UNIVERSIDAD DE BELGRANO

Las tesinas de Belgrano

Facultad de Ingeniería y Tecnología Informática

Carrera Ingeniería Informática

Integración de tecnologías Cloud Computing con Servicios Mobile: desarrollo de una aplicación del ámbito académico

N° 591

Carlos Sergio Rodrigo

Tutora: Mg. Paula M. Angeleri

Departamento de Investigaciones
2013

Universidad de Belgrano
Zabala 1837 (C1426DQ6)
Ciudad Autónoma de Buenos Aires - Argentina
Tel.: 011-4788-5400 int. 2533
e-mail: invest@ub.edu.ar
url: http://www.ub.edu.ar/investigaciones

Agradecimientos

La elaboración del presente Trabajo Final de Carrera ha sido una tarea importante en mi carrera y en mi vida, tanto personal y profesional. Por ello me gustaría aprovechar este espacio para agradecer a las personas que me ayudaron poder alcanzar esta meta.

A mi Tutora, que me guio para darle forma a una idea, marcar el camino, y hacerla posible.

A los docentes, que compartieron sus conocimientos y experiencia profesional en la búsqueda de formar profesionales.

A mis amigos que siempre creen en lo que soy capaz de alcanzar, incluso más que yo.

A mi Novia, que me ayudo a enfocar mis energías, supo escuchar mis frustraciones, y repetirme que se puede cuando más costaba.

A mi Madre y mi Hermana, que no dejaron de confiar en mí, darme energías y regaños en los momentos cuando el peso de esta laborar empezaba a sobrepasarme.

A mi Abuelo, por su apoyo en toda la carrera y la vida.

Índice

Índice de Figuras

1. Resumen

El presente trabajo final de carrera busca integrar conocimientos de distintas asignaturas mediante la construcción de una aplicación académica combinando las tecnologías de dispositivos móviles y *Cloud Computing*, y utilizando para su construcción una metodología de desarrollo basada en las metodologías ágiles de construcción de software, pero adecuándolas a las limitaciones de este proyecto.

Luego de una serie de intervalos de tiempo, se consigue una aplicación móvil para el sistema operativo Windows Phone que muestra información sobre las materias del plan de carrera de un alumno y su situación académica para con las mismas. La información acerca de las materias y el alumno, junto con un sitio web para la administración de esta información, está montada sobre Windows Azure.

La aplicación permite recibir notificaciones para mantener al alumno constantemente informado sobre su situación académica.

Junto con el presente Trabajo de Carrera se adjuntan en versión digital una serie de anexos que conforman el código fuente que fue generado durante su elaboración. Estos no se incluyen en el presente texto debido a la extensión de los mismos.

2. Organización del documento

El presente Trabajo Final de Carrera está organizado de la siguiente forma:

- **Introducción:** se describe el objetivo, justificación, alcance, y limitaciones del Trabajo Final de Carrera.
- **Marco Teórico:** se introducen los antecedentes y los conceptos fundamentales sobre los cuales se construye el Trabajo Final de Carrera. Estos comprenden Cloud Computing, Desarrollo ágil de software, Windows Phone, Windows Azure, entre otros.
- **Construcción:** en esta sección se realiza la construcción de la solución de software que busca construir el Trabajo Final de Carrera para cumplir con sus objetivos. Se aplica la metodología definida y se documentan los resultados obtenidos durante el desarrollo de la solución de software.
- **Conclusiones:** luego de obtener una solución de software, se analiza si su construcción contribuyo a alcanzar los objetivos definidos para el Trabajo Final de Carrera. También en esta sección se dispone a aportar futuras líneas de investigación para los estudiantes que estén finalizando sus estudios y quiera continuar la labor realizada en este trabajo.

3. Introducción

Mucho tiempo ha pasado desde la Maquina Analítica de Babbage, el primer hito para la revolución informática que hoy experimentamos, cuyo último hito son los teléfonos inteligentes y tabletas.

En la actualidad contamos con múltiples formas de acceder a nuestros sistemas informáticos, no solo en el tipo de cliente sino también en el dispositivo que lo contiene.

Los teléfonos inteligentes o Smartphone y las tabletas o Tablet han introducido un concepto muy poderoso y atractivo que es ***"Do anything anywhere"***, no solo permitiendo a los usuarios realizar tareas del ámbito laboral o personales, en un colectivo, un café, mientras miran televisión, entre otros entornos donde contar con una computadora es impensado o por lo menos incómodo, sino también cambian la forma en la que interactuamos con las aplicaciones. Esta es la clave de su rápida adopción en el mercado en general.

Dependiendo de la naturaleza de la solución informática, las capacidades de los actuales dispositivos bastan. Pero para aplicaciones comerciales o de la rama empresarial, donde se necesitan visualizar reportes de un gran volumen de datos, requieren de análisis de toneladas de información, los dispositivos móviles se ven muy acotados para satisfacer estas necesidades.

Es aquí donde entre en escena el otro actor principal del presente texto, Cloud Computing. Cloud Computing o la "Nube" como comúnmente se lo llama, es el nuevo paradigma de desarrollo de aplicaciones informáticas.

Cloud Computing propone externalizar los centros de datos de las organizaciones, para que estas puedan concentrarse en su *Business Core* (Su negocio principal, para una empresa metalúrgica seria la fabricación de productos metálicos). Obviamente esto exige un costo monetario, pero lo atractivo de la propuesta es que solo se cobra por el uso que se le da. Es decir, solo se abona la cantidad de megas de información, el tráfico que los sistemas tienen, y el procesamiento, por el uso que realmente se les da. De esta manera se evita soportar monetariamente grandes servidores, el servicios que consumen (luz, conexión a internet, etc.), el mantenimiento que requieren, y el espacio físico que los contiene. Mejorando la ecuación costo/beneficio.

Otro gran atractivo es la capacidad de escalar la infraestructura de manera dinámica acuerdo a la necesidad. Por ejemplo, si nuestros sistemas han incrementado su demanda en un 40% supongamos, para suplir la demanda se debería invertir en nuevos equipos y costos de implementación. Cloud Computing propone escalar dinámicamente las capacidades contratadas de acuerdo a la demanda, de manera que si se requiere más memoria para los sistemas, mayor capacidad de procesamiento, o de almacenamiento de información, solo se deberá realizar el pedido y el proveedor se encargará de incrementar las capacidades que habíamos contratado a las nuevas necesidades. También puede ser que los sistemas tenga un pico en su demanda de manera programada, y que luego vuelvan a su consumo normal, para este escenario también se puede escalar dinámicamente a las necesidades eventuales y luego volver a la configuración inicial.

Quizás pareciera que todo fuere ideal con la atractiva propuesta de la “Nube”, pero también es necesario hacer conciencia y, tener presente que toda nuestra información y la de nuestro negocio estará contenida en servidores de otra organización. Esta es la principal desventaja y argumento en su contra.

Por parte de los proveedores, se proponen contratos de confidencialidad y seguridad, que garantizan la integridad de los datos, con alto nivel de redundancia y mecanismos de backup’s de todos los tipos inventados, buscando de esta manera eliminar las dudas que puedan surgir al momento de contratar este servicio.

Los principales proveedores de Cloud Computing son Microsoft con *Windows Azure*, Google con *Google App Engine*, Salesforce, Amazon con *EC2 (Elastic Compute Cloud)*, Yahoo, y recientemente Apple con su *iCloud*.

Es importante visualizar que el valor de este nuevo paradigma no es solamente lo que los proveedores ofrecen, sino también la forma en que uno puede utilizar este servicio. Por ejemplo, como se trata más adelante, no solo podemos escalar en razón de infraestructura, sino también en prestaciones de los sistemas informáticos.

Para realizar esta investigación se elaborará una aplicación informática móvil para la Universidad de Belgrano, cuyo principal objetivo es el de brindar a alumnos herramientas para consultas y gestión de actividades académicas. En las próximas secciones se definirá formalmente el alcance de esta aplicación y como se llevará acabo.

Este escenario tiene la particularidad de que la demanda en el uso de la aplicación tiene variaciones durante el año lectivo, en época de exámenes tendrá una mayor demanda y luego un uso promedio. Lo que permitirá explotar una de las características fundamentales del Cloud Computing.

El investigador posee una marcada tendencia al uso de tecnologías Microsoft por la experiencia que posee trabajando con en ellas, por lo que la aplicación resultante de la investigación, estará construida bajo esta plataforma.

Con el objetivo de compartir la experiencia y conocimiento adquirido durante la elaboración de este trabajo, al momento de que el mismo esté finalizado se publicara todo lo desarrollado y las experiencias adquiridas, retribuyendo de esta manera a la comunidad IT y permitiendo la continuación y/o profundización de la problemática establecida en el trabajo.

3.1. Objetivo

3.1.1. Objetivo General

El presente Trabajo Final de Carrera (TFC) tiene como objetivo general integrar los conocimientos ingeniería de software, programación, base de datos, gestión de proyectos, en materias tales como Análisis de sistemas, Diseño de sistemas, Base de datos, Dirección de proyectos, Programación I,II, y III, Ingeniería de software avanzada.

El objetivo principal es el desarrollo de una aplicación de software de ámbito académico que utilice tecnologías emergentes.

3.1.2. Objetivos Específicos

Para poder cumplir con el objetivo principal se identifican los siguientes objetivos específicos:

- Utilizar una plataforma de Cloud Computing.
- Desarrollar un back-end de administración accesible vía web.
- Desarrollar un front-end para dispositivos móviles que es la tecnología más utilizada por los alumnos hoy en día.
- Establecer una metodología para la construcción de la solución considerando prácticas de metodologías agiles.

3.2. Justificación del trabajo

Una aplicación para dispositivos móviles permite al destinatario realizar uso de sus funcionalidades desde cualquier lugar. En un ambiente académico esto resulta de gran utilidad, por eso se busca conseguir una implementación del antiguo Sistema de Información a Alumnos (SIA) con el que contaba la Universidad de Belgrano para dispositivos móviles.

La complejidad que presenta el escenario de una aplicación académica, que a lo largo del año lectivo tiene picos de demanda y luego de estos, el acceso a la misma toma forma de una constante. Ese tipo de contexto es ideal para ser abordado con una arquitectura de Cloud Computing que permite escalar de manera ascendente o descendente.

El combinar ambas plataformas da como resultado una solución de software que permite brindar a los alumnos una aplicación móvil teniendo la seguridad que la arquitectura sobre la que está construida es capaz de soportar los picos de demanda sin afectar la experiencia del usuario.

3.3. Alcance

El alcance de esta aplicación, como se expondrá en la sección 5.Construcción, estará limitado a un sub conjunto de las funcionalidades del SIA de la Universidad de Belgrano adicionándose otras funcionalidades para aprovechar las ventajas de las tecnologías móviles. Dada la limitación en el tiempo de la presentación del presente TFC la aplicación móvil tendrá el siguiente alcance:

- Gestión de Materias
    - Visualización de Información de las materias
        - Fechas de Exámenes
        - Inscripción a Exámenes
        - Asistencia
- Envío de notificaciones al teléfono cuando:
    - Se vea comprometida la asistencia a una materia
    - Se aproxime una fecha de examen
    - Se aproxime el vencimiento de la fecha de inscripción a examen
    - Se publique el resultado de un examen

Del alcance definido, la aplicación móvil servirá como un cliente para visualizar la información de las materias, realizar acciones de inscripción a exámenes, y la visualización de las

notificaciones. Mientras que toda la lógica de negocio y el envío de las notificaciones se realizarán desde la aplicación web montada sobre una plataforma de Cloud Computing.

La aplicación web contará con las siguientes funcionalidades:

- **Agregar asistencias a un alumno**: se buscará un alumno por número de matrícula y código de facultad para agregar una asistencia, o una inasistencia.
- **Administración de Materias**: se buscará una materia por código de materia, una vez seleccionada se podrá establecer las fechas de examen parcial, examen recuperatorio, presentación final de trabajos prácticos, y examen final. Se podrá seleccionar a un alumno para completar la nota que obtuvo en un examen.

3.4. Limitaciones

La aplicación informática que se construirá durante el desarrollo del trabajo final de carrera estará desarrollada bajo tecnologías Microsoft, más específicamente, la plataforma de Cloud Computing será Windows Azure y el sistema operativo para teléfonos inteligentes donde se montará la aplicación móvil será Windows Phone v7.5 (Mango).

Esta decisión se basa en que el autor posee experiencia en el uso de las mismas, y se busca agilizar el desarrollo.

Tanto en el caso de Windows Azure como en el de Windows Phone, solo se desarrollará en la sección 4.Marco Teórico los aspectos que serán utilizados en la construcción de la aplicación a obtener en el Trabajo Final de Carrera, ya que ambas plataformas son extensas y poseen características que escapan a los objetivos de la misma.

La metodología elegida debe de poder ser utilizada por equipos pequeños de dos integrantes, donde solo uno de ellos tiene las competencias de programación. Mientras que el segundo integrante se encarga del liderazgo del proyecto y el control de calidad.

La búsqueda de antecedentes no será exhaustiva, sino que se centrará en buscar casos de estudios de la combinación de las plataformas Windows Azure y Windows Phone.

Queda excluida la elaboración de los manuales de usuario.

4. Marco Teórico

El marco teórico del presente trabajo final de carrera tratará de brindar al lector conocimientos en tres áreas.

Primero se buscará presentar Cloud Computing y como Microsoft con su plataforma Windows Azure implementa este paradigma.

Segundo, se introducirá a Windows Phone como sistema operativo para teléfonos inteligentes. Este es el sistema operativo de Microsoft para este tipo de plataformas.

Tercero y último, se describirán los origines del desarrollo ágil de software y las prácticas que se utilizarán para la construcción de la solución de software resultante del Trabajo Final de Carrera.

4.1. Antecedentes

Windows Phone y Windows Azure son plataformas de Microsoft, por lo que no resulta extraño pensar en combinarlas para crear aplicaciones que usen ambas tecnologías. Existen aplicaciones de este tipo de diversas clases, como es el caso de MicroFinance[1], una aplicación de finanzas montada sobre Windows Azure y con la posibilidad de ser accedida por Windows Phone.

Otro caso es el de News360[2], una emprendimiento en expansión que se dedica a brindar información de distinta índole pero de una manera personalizada al usuario. Este emprendimiento eligió Windows Azure para alojar el servicio y posee clientes en distintas plataformas móviles, entre ellas Windows Phone, Android, iPhone, etc.

La compañía telefónica T-Mobile[3] combinó ambas tecnologías para conseguir una solución que sirva a las familias para establecer un canal de comunicación sin utilizar las redes sociales. Este caso de estudio además expone la velocidad con la que el equipo de desarrollo pudo conseguir el resultado alcanzado.

---

[1] *MicroFinance Case Study http://blogs.technet.com/b/uktechnet/archive/2012/01/17/windows-azure-case-study-mando-group-microfinance-app.aspx. Fecha de acceso: 18/03/2013*

[2] *News360 Case Study http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?casestudyid=710000000283. Fecha de acceso: 18/03/2013*

[3] *T-Mobile Case Study http://www.microsoft.com/casestudies/Windows-Azure/T-Mobile-USA/Mobile-Operator-Speeds-Time-to-Market-for-Innovative-Social-Networking-Solution/4000008598. Fecha de acceso: 18/03/2013*

Sin embargo durante la búsqueda realizada no se encontró un caso de estudio acerca de combinar ambas plataformas para la construcción de una aplicación académica.

4.2. Cloud Computing

Cloud Computing es el nuevo paradigma para construcción de aplicaciones distribuidas. Posee distintas características que pueden adaptarse a negocios de grandes o pequeñas escalas, como ya se adelantó en la introducción, estos negocios solo pagaran por el uso que le dé a estas características.

Sriram Krishnan en (Krishnan, 2010, p. 1) provee de una gran base de conocimiento sobre Windows Azure, la plataforma de Microsoft para Cloud Computing, y da una muy buena definición para entender este concepto:

"Imagínese si no existiera el agua del grifo. Cada hogar tendría que cavar un pozo, y eso sería un trabajo arduo. Los Pozos son caros construir y caros de mantener. No sería capaz de obtener una gran cantidad de agua rápidamente si lo necesita, al menos no sin actualizar su bomba. Y si ya no necesita el pozo o la bomba, no tendría q quien devolverlo, y no podría recuperar su inversión inicial. Si usted dejó vacante la casa, o la estructura adecuada que instalaron en su casa, habría invertido en un pozo que no necesita.

El Agua del grifo soluciona todo eso. Alguien gasta el dinero y construye la plomería correcta y la infraestructura. Alguien se encarga de administrarla y, que el agua sea limpia y siempre disponibles. Usted paga sólo por lo que utiliza. Siempre puede obtener más si lo desea.

En resumen, es lo que Cloud Computing es. Son recursos de un Data Center entregados como agua del grifo. Siempre están disponibles, y usted paga sólo para lo que utiliza."

Dentro de Cloud Computing existen servicios que engloban usos particulares para este paradigma, ***Infrastructure as a Service (IaaS)***, ***Plataform as a Service (PaaS)***, y ***Software as a Service (SaaS)***.

*"**Infrastructure as a Service (IaaS)**: Esta categoría se refiere a los servicios que proveen los niveles más bajos de la pila. Generalmente brindan hardware básico como un servicio, cosas como Máquinas Virtuales, configuración de balanceo de carga, y almacenamiento en la Red.*

Plataform as a Service (PaaS)*: En este caso, el termino plataforma se refiere a una forma de abstraer las los niveles más bajos de la pila. Esta aplicación se ejecuta en un ambiente especializado. Y este ambiente es a veces restringido (ejecutándose con privilegios de bajo nivel, con restricciones para escribir en el disco local, y limitaciones de ese tipo). Los proveedores brindan también, abstracciones para servicios básicos (como Email, cache distribuido, almacenamiento estructurado), y proveen enlaces para varios lenguajes. En el caso de Windows Azure, típicamente las aplicaciones se escriben en .Net, pero también se pueden escribir en código nativo, escribir código en otros lenguajes y runtimes como Python/Ruby/PHP/Java, y, en general, ejecutar la mayoría de código que se puede ejecutar en Windows.*

Software as a Service (SaaS)*: El ejemplo canónico de este modelo es Salesforce.com. En este caso, las aplicaciones pueden accederse desde cualquier lugar. En lugar de alojar aplicaciones como Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), y Human Resources (HR) de manera on-site, las compañías pueden externalizar estas aplicaciones."*
(Krishnan, 2010, p. 8)

4.2.1.Plataforma Windows Azure

Windows Azure es la plataforma de Microsoft para Cloud Computing, está compuesta por las siguientes tecnologías: *Windows Azure*, *Azure AppFabric*, y *SQL Azure*.

De estos componentes se hará foco principal sobre *Windows Azure*, ya que este es el que se encarga ejecutar las aplicaciones en la nube. Sobre el resto, *Azure AppFabric* y *SQL Azure*, se dará una pequeña reseña ya que están fuera del alcance del presente TFC. Antes de empezar a desarrollar los componentes es necesario aclarar que cada componente es independiente, si bien una aplicación puede estar montada sobre *Windows Azure* y usar los servicios *de Azure AppFabric* y *SQL Azure*, también una aplicación puede estar alojada en el datacenter propio de una organización y usar los mismos servicios. Es decir, no es necesario que la aplicación este montada sobre Windows Azure para consumir el resto de los componentes de la plataforma.

4.2.1.1. Azure AppFabric

Esta tecnología provee servicios típicos de infraestructura y control de acceso a aplicaciones que pueden estar montadas sobre Windows Azure o no. Esto es posible ya que expone sus servicios mediante una HTTP REST API.

Los componentes de esta tecnología son:

4.2.1.1.1. Service Bus

Permite exponer WCF endpoints que pueden ser accedidos por cualquier tipo de aplicación como HTTP Url desde cualquier parte. De esta manera podemos comunicar aplicaciones que se encuentran en distintas redes sin tener que lidiar con firewalls, hardware, direcciones de red, entre otros problemas comunes de esta clase de escenarios.

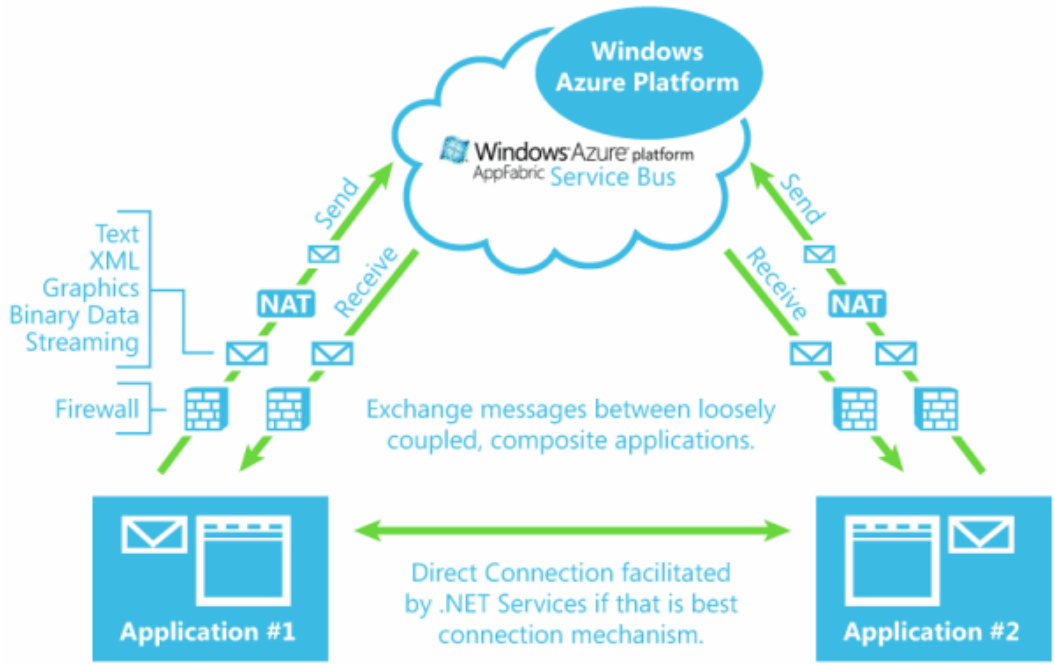

Figura 1 - Esquema Service Bus[4]

4.2.1.1.2. Access Control

Este servicio permite utilizar *autenticación federada*[5] mediante un modelo REST, sin tener que abordar demasiado en este modelo, solo definiendo reglas sobre las cuales dejaremos que acceder a nuestras aplicaciones o no.

También permite la integración con *Active Directory Federation Services*[6].

[4] *Service Bus* http://i.msdn.microsoft.com/gg318629.service-bus(es-es,MSDN.10).gif. *Fecha de acceso: 18/03/2013*

[5] *Identidad o Autenticación Federada http://es.wikipedia.org/wiki/Identidad_federada.Fecha de acceso: 18/03/2013*

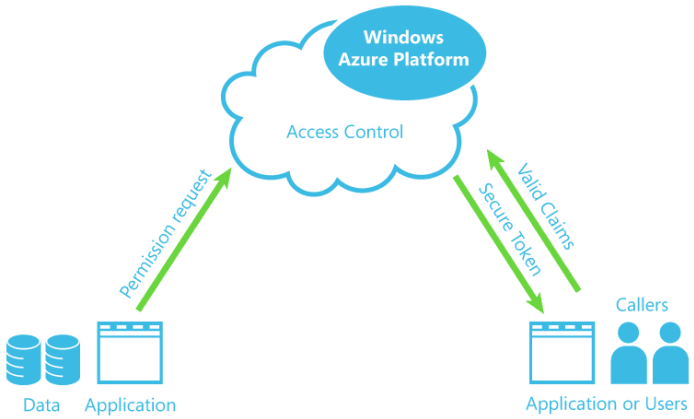

Figura 2 - Esquema de Access Control[7]

4.2.1.2. SQL Azure

SQL Azure básicamente es un servicio que nos permite utilizar una o varias instancias (según se necesite) de SQL Server, contaremos con una base de datos relacional clásica de SQL Server 2008 R2 montada en la nube, con capacidad de replicarse y bajo la premisa de solo pagar lo que se consume. Evitándonos de esta forma el costo de licencias, y compra y mantenimiento de hardware.

En el presente TFC no se utilizará este sistema de almacenamiento de datos, ya que se optó por utilizar los servicios de almacenamiento que provee la tecnología Windows Azure.

4.2.1.3. Windows Azure

Windows Azure es la tecnología dentro de la Plataforma Windows Azure para montar aplicaciones en la nube. Nuestras aplicaciones contaran con capacidad de escalar, ser hospedadas en los data centers de Microsoft distribuidos alrededor del mundo (con la opción de poder elegir donde queremos hospedarlas), replicación de datos, capacidad de almacenamiento, y todo esto bajo el modelo de solo pagar por el uso que le damos.

---

[6] *Active Directory Federation Services http://technet.microsoft.com/es-es/library/adfs2(v=ws.10). Fecha de acceso: 18/03/2013*

[7] *Windows Azure Access Control http://i.msdn.microsoft.com/gg318629.access-control(es-es,MSDN.10).gif. Fecha de acceso: 18/03/2013*

Windows Azure no es otro sistema operativo de Microsoft que podemos instalar en nuestra computadora, sino que está distribuido entre cientos de máquinas. Posee la capacidad para monitorear el hardware bajo su dominio para avisar de fallas en caso de no poder hacerles frente por sí mismo, como así también generar nuevas instancias de nuestra aplicación en caso de que necesitemos escalar porque la demanda aumentó.

4.2.1.3.1. Características

Se han nombrado varias características con el objetivo de introducir Windows Azure, a continuación se dispone a profundizar sobre ellas:

4.2.1.3.1.1. Hospedaje de Servicios

Cuando hablamos de Servicios, nos referimos en este caso a todo tipo de aplicación que se ejecuta del lado del servidor. Windows Azure permite hospedar Sitios Web, rutinas de procesamiento de datos, entre otro tipo de aplicaciones. Si bien es simple, para poder alojar nuestras aplicaciones deben cumplir con el *Service Model*, el que se explica con mayor detalle en esta misma sección.

4.2.1.3.1.2. Administración de Servicios

En un datacenter propio, cuando necesitamos la actualización de un parche del sistema operativo, remplazar hardware averiado, debemos lidiar con estas tareas e incluso en ocasiones interrumpir la ejecución de nuestra aplicación. Windows Azure se encarga de esto de una manera transparente para nosotros, sin tener que interrumpir la ejecución de nuestra aplicación. La pieza encargada de esto es el *Fabric Controller*, definido por (Krishnan, 2010) como:

"El Fabric Controller es llamado también "el cerebro" de Windows Azure, por una buena razón: él controla las operaciones de todas las otras máquinas, así como también los servicios que son hospedados en ellas". (Krishnan, 2010, p. 35)

El *Fabric Controller* monitorea todas las maquinas, y cuando se producen fallas de hardware las gestiona (por sí mismo o avisa al personal de mantenimiento), y si nuestra aplicación deja de tener servicio se encarga de activar una nueva instancia y ponerla disponible nuevamente. Para esto último se utiliza otro componente de la arquitectura de Windows Azure: el *Hypervisor*.

Si bien Windows Azure está montado sobre miles de servidores físicos, la Virtualización juega un papel fundamental en un servicio de Cloud Computing. Para poder gestionar el hardware

físico y proveer de instancias para las distintas necesidades de manera eficiente, Windows Azure cuenta con el *Hypervisor* para la gestión de recursos como CPU y acceso I/O entre las máquinas virtuales y el aislamiento entre ellas para evitar colisiones. Básicamente orquesta las distintas máquinas virtuales de los servidores.

Estos componentes son de suma importancia para la plataforma de Windows Azure, pero no se profundizará más en ellos debido a que están fuera del alcance del presente TFC, como se estableció en la sección 3.4. Limitaciones.

4.2.1.3.1.3. Almacenamiento

Windows Azure provee de tres estructuras de datos cada una con un fin específico:

- ***Blobs***: *Binary Large Objects* es una estructura de datos dedicada a almacenar grandes cantidades de información, como videos, imágenes de grandes resoluciones, etc.
- ***Queues***: son colas para almacenar llamadas o tareas, se utilizan generalmente desacoplar componentes de la aplicación, por nombrar un uso.
- ***Tables***: son tablas semi estructuradas de datos, a diferencia de SQL Azure, estas tablas no responden la paradigma relacional de bases de datos.

Estas estructuras de datos serán utilizadas en la implementación de la aplicación que dará como resultado el TFC, por lo que en se realiza una descripción más profunda al final de esta sección.

4.2.1.3.1.4. Virtualización

Nos permite obtener máquinas virtuales en cuestión de minutos o incluso segundos. Estas máquinas pueden generarse con los siguientes sistemas operativos:

- Windows Server 2008 SP1
- Windows Server 8 Beta
- CentOS 6.2
- OpenSUSE64 1.2.1 Beta
- SUSE Linux Enterprise Server
- Ubuntu Server 12.04

4.2.1.3.1.5. Múltiples plataformas de desarrollo

Windows Azure provee a la comunidad de desarrolladores SDKs para distintas plataformas, además de Microsoft .NET. Estos SDKs son *Open Source* licenciados bajo la licencia Apache 2 y mantenidos en repositorios de GitHub[8].

Los lenguajes que poseen SDK son .net, node.js, php, java, y phyton.

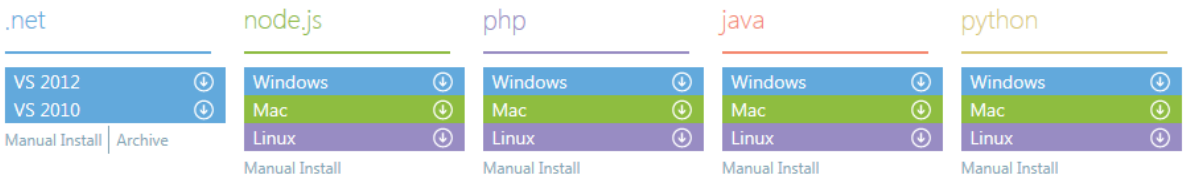

Figura 3 - Windows Azure SDKs (Windows Azure, 2012)

4.2.1.3.2. Service Model en Windows Azure

Como se mencionó anteriormente, en Windows Azure toda aplicación que se desea montar sobre la plataforma es considerada un servicio. Para poder ser montada y contar con las capacidades que la plataforma ofrece debe responder a un modelo, el mismo establece que cada parte de la aplicación debe de responder a un Rol de Windows Azure, y permite generar un paquete con todo lo necesario para montar el servicio.

4.2.1.3.2.1. Windows Azure Roles

Los *Windows Azure Roles*, no se refieren a un rol de usuario como puede ser el de Administrador o Usuario, sino que están orientados al tipo de aplicación que será montada en Windows Azure.

Un servicio puede estar compuesto por múltiples Roles, donde uno se encarga del front-end web de la aplicación y otro del análisis de datos, por dar un ejemplo. Básicamente existen dos tipos de roles, el *Web Rol* dedicado a sitios web y web services, y el *Worker Rol* orientado a aplicaciones de procesamiento de datos tales como una consola, o un servicio de Windows por ejemplo.

Los Roles son el tipo de aplicación, que pueden tener varias instancias del mismo. Para ayudar a entender los roles, Siriam Krishman (Krishnan, 2010, p. 67) proporciona una analógica

[8] *GitHub http://www.github.com. Fecha de acceso: 18/03/2013*

sumamente didáctica. Propone pensar en Windows Azure y los Roles como el uso de un pizarrón para diseño de la arquitectura de un sistema. Se dibujan cajas que pueden representar un sitio web u otro componente y se interrelacionan con flechas, básicamente Windows Azure es el pizarrón y las distintas cajas son roles del servicio, interconectados mediante puntos de acceso. Como sucede en la vida real, puede que un componente del sistema necesite de más de un servidor debido a que el uso del mismo será extensivo, en Windows Azure podemos definir cuantas instancias tendrá un Rol determinado.

El servicio cuenta con dos archivos que sirven para definir las características del servicio y la configuración del mismo, estos son el *Service Definition* y el *Service Configuration*.

4.2.1.3.2.2. Service Definition

Este es un archivo XML cuyo objetivo es definir los roles que componen al servicio y sus características. En él se definen:

- Roles y sus configuraciones
  - **Endpoints**: Puntos de entrada al rol, pueden ser internos (para intercomunicación entre roles) mediante un InternalEndpoint o para comunicarse con el exterior mediante InputEndpoint. En el último caso se asigna el puerto que el rol estará escuchando.
  - **Tamaño del Rol**: se refiere a las características de la máquina virtual sobre la que estará montado el Rol.
    - ExtraSmall – CPU: 1.0GHz, RAM: 768MB, HDD: 20GB.
    - Small – CPU: 1.6GHz, RAM: 1.75GB, HDD: 225GB.
    - Medium – CPU: 2x1.6GHz, 3.5GB, HDD: 490GB.
    - Large – CPU: 4x1.6GHz, 7GB, HDD: 1000GB.
    - ExtraLarge – CPU: 8x1.6GHz, 14GB, HDD: 2040GB
  - **Configuraciones que va a usar el Rol**: En este archivo solo se declara el nombre de las configuraciones que va a utilizar el Rol, la configuración se define en el archivo *Service Configuration*.

Un ejemplo de este archivo es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<ServiceDefinition name="WindowsAzureEjemplo"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2012-
05.1.7">
  <WebRole name="WebRoleEjemplo" vmsize="Medium">
    <Sites>
      <Site name="EjemploTesinaUB">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <ConfigurationSettings>
      <Setting name="ConnectionStringWebRol"/>
    </ConfigurationSettings>
  </WebRole>
  <WorkerRole name="WorkerRoleEjemplo" vmsize="Small">
    <ConfigurationSettings>
      <Setting name="ConnectionStringWorkerRol"/>
    </ConfigurationSettings>
  </WorkerRole>
</ServiceDefinition>
```

Es importante destacar que si se hacen cambios en este archivo es necesario volver a generar el paquete del servicio y actualizar el servicio en Windows Azure para que los cambios se hagan efectivos.

4.2.1.3.2.3. Service Configuration

Este archivo también es un archivo XML pero con el propósito de definir:

- **Cantidad de instancias del Rol**: se define el número de instancias que el Rol tendrá en ejecución.

- **Valores de Configuraciones**: en el archivo Service Definition, se incluyen los nombres de las configuraciones que el Rol puede usar, en el Service Configuration se definen estas configuraciones, como un conjunto nombre – valor.

Ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>

<ServiceConfiguration serviceName="WindowsAzureEjemplo"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration" osFamily="1"
osVersion="*" schemaVersion="2012-05.1.7">
  <Role name="WebRoleEjemplo">
    <Instances count="5" />
    <ConfigurationSettings>
      <Setting name="ConnectionStringWebRol" value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
  <Role name="WorkerRoleEjemplo">
    <Instances count="2" />
    <ConfigurationSettings>
     <Setting name="ConnectionStringWorkerRol" value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

Este archivo, a diferencia del *Service Definition*, si permite ser modificado sin necesidad de volver a generar el paquete del servicio, por lo que podemos cambiar las configuraciones y estas serán aplicadas en el próximo pedido que reciba el servicio.

4.2.1.3.2.4. Empaquetado del Servicio

Para montar los servicios sobre Windows Azure se construye un paquete que contiene los binarios de la aplicación y los archivos *Service Definition* y *Service Configuration*. Este paquete son dos archivos, uno con extensión *cspkg* que contiene los binarios del servicio y las definiciones del mismo. Y un segundo archivo llamado *ServiceConfiguration.cscfg*, que permite hacer cambios en la configuración del servicio sin tener que volver a generar el paquete o actualizar el que ya se

encuentra en la nube. El servicio puede ser montado a la plataforma de distintas formas, desde Visual Studio, desde el Portal de Administración de Azure, desde GitHub, entre otras.

4.2.1.3.3. Storage en Windows Azure

El Storage o almacenamiento en Windows Azure, es un servicio. Cada tipo de almacenamiento permite operar sobre él mediante una Api, para lo cual solo necesitaremos las credenciales de nuestra cuenta de servicio.

Como ya se ha comentado, existen varias alternativas para el almacenamiento de información en la plataforma de Windows Azure, ambas alternativas comparten características sumamente atractivas, según (Krishnan, 2010, pp. 129,130) estas son:

- **Escalabilidad**: todos los servicios de almacenamiento de datos son escalables, podemos incrementar la cantidad de datos sin preocuparnos por límites de hardware en los servidores, performance, y mantenimiento.
- **Replicación**: toda la información almacenada es replicada múltiples veces y distribuida en distintos servidores, para que en caso de fallas no perdamos información y continúe estando a nuestra disposición garantizando un *downtime* prácticamente nulo.
- **Geo distribución**: cuando se crea una cuenta de almacenamiento de información, se puede elegir en que región geográfica se almacenará, esto mejora la performance alojando la información cerca de donde están montadas las aplicaciones. Y además permite evitar que en caso de un desastre se concrete en una región no afecte nuestra información.
- **Pagar solo por el uso**: como el resto de los servicios de Cloud Computing, solo se paga por el almacenamiento usado.
- **RESTFul HTTP APIs**: todas las estructuras de datos ofrecidas por Windows Azure pueden ser consumidas mediante APIs RESTFul HTTP, esto nos permite construir librerías o consumir nuestra información desde distintas aplicaciones.
- **Consistencia**: El almacenamiento en Windows Azure es inmediatamente consistente, esto quiere decir que cada cambio será impactado de inmediato, y las siguientes consultas tendrán se realizarán sobre el nuevo estado de la información.
- **Mucho espacio**: nos liberamos de la preocupación sobre cuanto espacio disponible tenemos, Windows Azure soporta increíbles cantidades de información en sus estructuras (se hablará de ello en las próximas secciones).

4.2.1.3.3.1. Blobs

Los Blobs o *Binary Large Objects* es una de las alternativas respecto a almacenamiento en la nube. Puede soportar grandes cantidades de información desestructurada junto con su metadata. Al decir “grandes cantidades”, nos referimos al orden de miles de Gigabytes de información, que es replicada múltiples veces para evitar pérdidas por fallas de hardware.

Dentro de ellos es posible almacenar imágenes, videos, música, archivos cartográficos, por nombrar ejemplos de tipos de información que suelen ocupar grandes cantidades.

Los Blobs están alojados en *Containers*, que pueden ser públicos o privados, y permiten alojar más de un Blob.

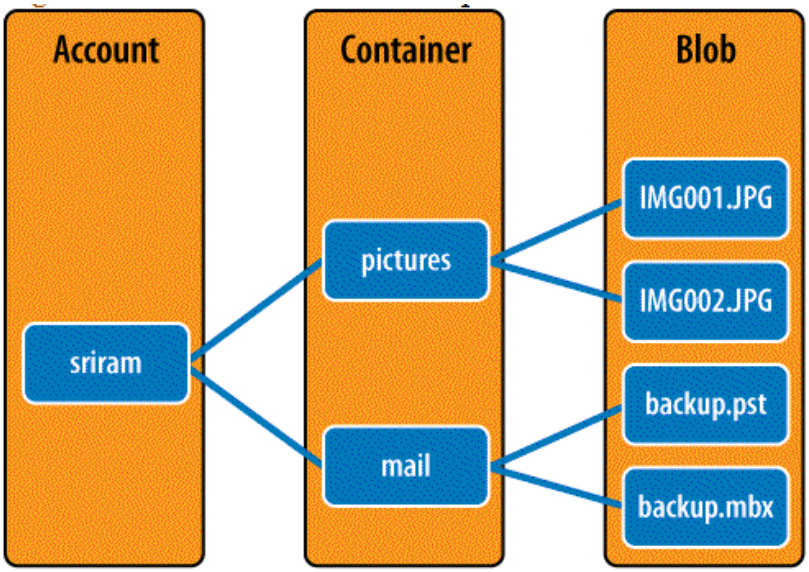

Figura 4 - Blobs (Krishnan, 2010, p. 160)

Como se comentó anteriormente, todos los servicios de almacenamiento en Azure pueden ser consultados mediante una API REST HTTP, de manera que podríamos almacenar información en Windows Azure y consumirla desde una aplicación montada en un servidor propio si se lo desea. Y si el *Container* es público, podría ser consultado por cualquier usuario de Internet.

4.2.1.3.3.2. Tables

Las Tables o Tablas, son otra forma de almacenar información en la plataforma de Windows Azure. A diferencia de los Blobs, las Tables almacenan información estructurada en forma de *Entidades* y cada entidad tiene *Propiedades*.

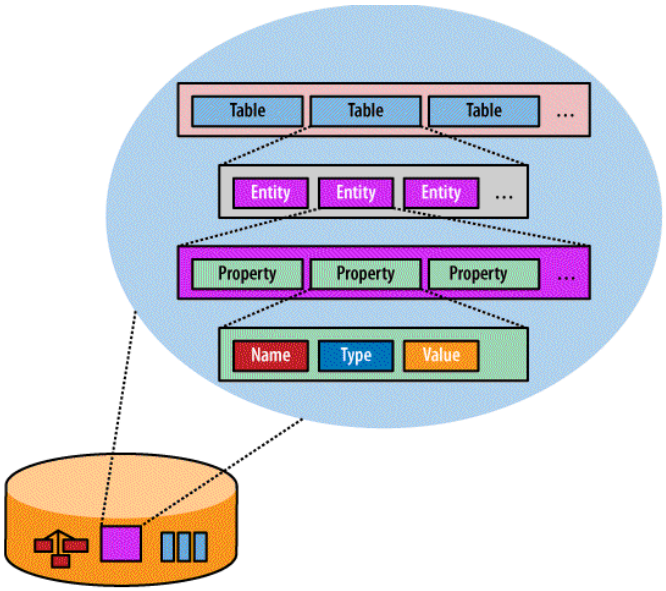

Figura 5 - Tables (Krishnan, 2010, p. 226)

Básicamente podríamos establecer un paralelismo con una Tabla del Modelo Relacional de Base de Datos, pero la principal diferencia se encuentra en que la información en el modelo de Tablas de Windows Azure no se relaciona con otras tablas. Las entidades pueden cambiar y no tendremos la necesidad de luchar contra restricciones de Claves Foráneas o volver a generar tablas cuando se agrega una nueva propiedad a nuestra entidad, además nos libraremos del costo de performance y esfuerzo que tienen el mantener normalizada una base de datos de gran tamaño.

Las desventajas de esto es que la integridad de la información debe ser gestionada por la aplicación y no por el motor de base de datos.

Las tablas proporcionan un almacenamiento que puede escalar sin límites, pero toda esa información va a estar alojada en múltiples máquinas. Entonces surge la pregunta, ¿Qué tan

eficientes pueden ser las consultas sobre mis datos si tienen que recorrer múltiples nodos para recoger toda la información?, para esto se utiliza el concepto: *Particionamiento*.

El *Particionamiento* se refiere a como agrupar las entidades, para que de esta forma se optimicen las consultas y la información se aloje en el mismo servidor o nodo de servidores. Para hacer posible esto, cada entidad tiene, además de su conjunto de propiedades, dos propiedades más:

- **Partition Key**: es la clave que agrupa las entidades en una misma partición, es decir, todas las entidades con la misma clave estarán agrupadas en la misma partición.
- **Row Key**: es la clave que identifica a la fila.

La PartitionKey + RowKey en conjunto forma la Clave Primaria de la entidad.

4.2.1.3.3.3. Queue

El *Queue Storage Services*, o servicio de almacenamiento de Colas es un servicio de colas de mensajes que provee la plataforma de Windows Azure, en el mismo podrán encolarse incluso millones de mensajes.

Las colas sirven para distintos propósitos muy diversos, por ejemplo la implementación de un procesador sintáctico de texto, o de operaciones matemáticas, pero en el presente TFC se les dará el uso que comúnmente se les da en una aplicación montada en la Nube, *servir de ayuda para realizar operaciones asincrónicas*. A partir de ellas podremos encolar mensajes para que otros componentes de nuestro sistema realicen diferentes acciones, desacoplando así funcionalidades.

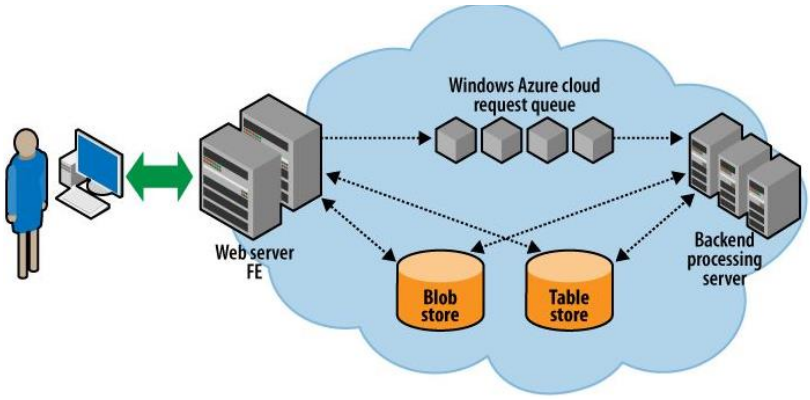

Figura 6 - Queues (Krishnan, 2010, p. 205)

Desacoplar los componentes de nuestra aplicación como sugiere la imagen supone una gran ventaja, ya que si alguna falla, la aplicación sigue funcionando y la experiencia de usuario no se ve afectada. Para explicar un poco mejor supongamos un simple blog, separado en front-end y back-end comunicados mediante una cola de mensajes. Si el back-end falla cuando se hizo un comentario en un post, el front-end actualizará el post mostrando el comentario y enviará el mismo al back-end depositando la acción en la cola. Como el back-end no está activo, el mensaje no es procesado pero no se pierde, y la experiencia de usuario es la misma que si el blog funcionara correctamente. Cuando el back-end reanude sus tareas procesará todos los comentarios en la cola. Si se toma como ejemplo el escenario donde el back-end falla al momento de procesar un comentario de un post, el servicios de Colas de Windows Azure tiene un mecanismo que los ítems deben ser marcados como procesados para eliminarse de las colas, por lo que si el back-end falla en medio de una operación, el ítem que se estaba procesando volverá a aparecer en la cola para volver a realizar la operación.

Un uso de las mismas que se explicará en la sección de 5.Construcción, es el de depositar notificaciones para ser enviadas a los teléfonos inteligentes.

4.3. Windows Phone

Windows Phone es el sistema operativo para teléfonos inteligentes de Microsoft. Presentado el 15 de febrero del 2010, representa un nuevo capítulo para Microsoft en materia de sistemas operativos para dispositivos móviles, separado totalmente de su antecesor Windows Mobile.

Este sistema operativo trata de diferenciarse totalmente de sus competidores con una Interfaz Gráfica diferente e innovadora llamada en principio Metro UI, y recientemente renombrada a *Modern UI*. Para desarrollar aplicaciones que se ejecuten en el mismo, se utiliza un SDK que Microsoft pone a disposición de los desarrolladores y la versión de 2.2.1.Silverlight para Windows Phone. Las aplicaciones al igual que los juegos se distribuyen mediante el *Marketplace* de aplicaciones, que para publicarlos se los evalúa en aspectos de performance, memoria, características del teléfono que utiliza (GPS. Cámara fotográfica, conexión a internet, etc.), para que no se puedan descargar aplicaciones que dañen los equipos y proteger a los usuarios.

En la actualidad se encuentra en la versión 7.5 “Mango”, que incorporó funcionalidades como la capacidad de ejecutar múltiples aplicaciones a la vez, Internet Explorer 9, integración con

las redes sociales Twitter y Linkedin, entre otras. Será bajo esta versión sobre la que se realizará el TFC.

Los aspectos que se exponen de Windows Phone será los usados en la construcción de la aplicación a desarrollar en el TFC, existen otros además de estos, pero como se expuso en la sección 3.4.Limitaciones, el resto de ellos quedan fuera del alcance.

4.3.1.Silverlight para Windows Phone

Las aplicaciones en Windows Phone se desarrollan sobre una versión de Silverlight 4 para Windows Phone, ésta es una versión reducida y adaptada para el sistema operativo móvil.

Silverlight permite crear aplicaciones con Interfaces de Usuario realmente atractivas, y la versión para Windows Phone trata de mantener la línea de diseño que propone *ModernUI*, logrando en conjunto aplicaciones que generan una gran experiencia de usuario.

Las aplicaciones en Windows Phone se conforman de una o varias páginas, llamadas *PhoneApplicationPage's*. Las páginas tienen una *vista* que se escribe en un lenguaje llamado XAML, y *code-beside* que es el lugar donde se codifican los eventos que se disparan desde la vista, como puede ser presionar un botón o completar un cuadro de texto. Esta parte puede ser codificada en C# o en VisualBasic, en el presente TFC se eligió el uso de C# porque se posee experiencia previa en este lenguaje.

El SDK de Windows Phone nos provee de tres plantillas de aplicaciones, que se diferencian principalmente en la manera que muestran la información. La primera es una aplicación con una página vacía, la segunda es una aplicación Panorama, y la tercera es una aplicación Pivot.

Estas últimas dos plantillas son básicamente páginas con un Control de Silverlight que sirve de contenedor para múltiples páginas, y se diferencian en cómo estos controles muestran las páginas que contienen.

4.3.1.1. Control Panorama

Como se mencionó anteriormente, el control Panorama permite contener varias páginas. La particularidad de este control es que muestra las páginas de manera horizontal lo cual lo convierte en una excelente opción para presentar las características de una aplicación.

Para mostrar su funcionalidad se tomará el ejemplo presentado en *Central Application Hub with Home Page Menu (Panorama or Pivot Control) for Windows Phone*[9].

Figura 7 - Ejemplo de Panorama Control

La figura muestra la aplicación de música y video que viene pre-instalada en Windows Phone, el control Panorama posee varios paneles o paginas cada uno con un objetivo particular, y permite agregar una imagen de fondo para conseguir un diseño atractivo.

Se elige una página inicial y luego el usuario puede navegar hacia la derecha e ir descubriendo más contenido.

Cada página no solo puede mostrar información, sino que también puede dirigir a otras páginas para realizar otras tareas. Esto significa que una aplicación puede cambiar de Interfaz de Usuario según la necesidad y no se limita solo a la que brinda el Panorama. Si se seleccionase el título “Radio” en el primer panel que se ve de la Figura 7, la aplicación se dirigiría a la aplicación de Radio.

[9] *Central Application Hub with Home Page Menu (Panorama or Pivot Control) for Windows Phone* http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202892(v=vs.92).aspx. *Fecha de acceso: 18/03/2013*

Figura 8 - Aplicación Radio

4.3.1.2. Control Pivot

Este control propone una experiencia de usuario del estilo de Pestañas. Se recomienda usar cuando la aplicación está enfocada en un tipo de información en particular, y cada pestaña actuará como filtro. Cada pestaña actúa como si fuera la primera página de la aplicación, por lo que si se presiona el botón de atrás del teléfono, la aplicación se cierra.

Una práctica recomendada, que depende del tipo de aplicación, es que se use un Panorama al inicio de la aplicación y luego a un Pivot.

Figura 9 - Pivot Control - Aplicación Outlook

4.3.2. Ciclo de Vida de una Aplicación

Una aplicación en Windows Phone pasa por distintos estados, por ejemplo cuando se inicia, cuando se cierra, y cuando se suspende para que el usuario realice otras tareas.

El modelo de ejecución de aplicaciones en Windows Phone solo admite una aplicación ejecutándose a la vez, buscando economizar los recursos como la batería del teléfono o el consumo de datos. Sin embargo, logra dar la sensación de que se están ejecutando múltiples aplicaciones a la vez mediante el cambio de estados y la persistencia de información a través de ellos.

Esta información se puede guardar de dos formas, *Persistente* y *Transitoria*. La primera tiene como objetivo mantener información relacionada a configuraciones y datos con los que la aplicación siempre debería contar. Se guardan en el *IsolateStorage*, que es una sección privada de almacenamiento dedicada a la aplicación, y cada aplicación solo puede acceder a su *IsolateStorage*. La segunda, tiene como objetivo mantener información de la aplicación en caso de que esta sea suspendida, de manera que al volver a la misma genera la sensación de que la aplicación siguió ejecutándose cuando estuvo en segundo plano. Se puede guardar información de manera transitoria a nivel de aplicación y de página. A nivel de aplicación la información se guarda en el diccionario *PhoneApplicationService*, que se instancia al momento que la aplicación empieza a ejecutarse, y solo debe usarse para guardar información relativa a la aplicación como un todo. En el caso de las paginas, y así como el *PhoneApplicationService*, al momento de iniciar la aplicación se instancia un diccionario *PhoneApplicationPage* por cada página con la que cuente la aplicación.

Como se explica en el artículo *Execution Model Overview for Windows*[10], una aplicación en Windows Phone puede pasar por tres estados: *Running*, *Dormant*, y *Tombstoned*. Y transitar por estos estados se disparan distintos eventos, que además de realizar el cambio de estado, son los que permiten guardar información sobre la aplicación como se comentó recientemente. En la Figura 10 se puede ver el modelo de ejecución completo de una aplicación en Windows Phone.

[10] *Execution Model Overview for Windows* http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008(v=vs.92).aspx. *Fecha de acceso: 18/03/2013*

4.3.2.1. Evento *Launching*

Este evento es disparado cuando se quiere iniciar la aplicación desde el Tile en el Start Experience, desde la lista de aplicaciones, o a partir de una 4.1.2.2.Notificaciones Toast. Al iniciarse la aplicación de este modo, se genera una nueva instancia de la misma. Es una buena práctica que no se use este evento para realizar ninguna tarea pesada, como obtener datos de internet o procesar grandes cantidades de información, ya que uno de los requerimientos para la certificación de aplicaciones en Windows Phone es que la aplicación no puede tardar más de 5 segundos en iniciar.

4.3.2.2. Evento *OnNavigatedTo*

Este evento se dispara cuando el usuario navega a una página de la aplicación. Esta situación se produce al iniciar la aplicación, al utilizar la aplicación y navegar por ella, o cuando la aplicación vuelve a iniciar luego de estar en un estado *Dormant* o *Tombstoned*, que se explican en las próximas secciones.

Siempre que este evento se dispara, es una práctica recomendada restaurar información almacenada en los diccionarios del *PhoneApplicationService* y *PhoneApplicationPage* de cada página. Sin embargo, antes de realizar esta tarea se debe de comprobar que se está trabajando con instancias nuevas de las paginas, de modo contrario no es necesario restaurar la información de las mismas porque el sistema operativo se encargará de ello.

4.1.1.1. Estado *Running*

La aplicación se encuentra en estado *Running* luego de que es iniciada, y permanece en este estado durante todo el tiempo que se navegue en ella.

La aplicación sale de este estado cuando se produce una interrupción en el uso de la misma, como puede ser el bloqueo del teléfono, la activación de otra aplicación mediante el uso del botón de inicio y la posterior elección de otra aplicación, o a partir de una *Toast Notification*.

Al momento de producirse la interrupción, se disparan otros eventos (que explican en las secciones siguientes) y la aplicación pasa a un estado *Dormant* o *Tombstoned*.

4.1.1.2. Evento *OnNavigatedFrom*

Así como el evento *OnNavigatedTo* se dispara cuando el usuario navega hacia una página de la aplicación, el evento *OnNavigateFrom* lo hace cuando se abandona una página de la aplicación.

Cuando se dispara el evento es recomendable guardar información de la página de manera *Transitoria* utilizando el diccionario *PhoneApplicationPage* que le provee el sistema operativo a la página de la aplicación, y así poder recuperar esa información cuando se vuelva a la página luego de una interrupción en el uso de la aplicación, ya que como se describió anteriormente, el sistema operativo puede realizar esta tarea si la página no es una instancia nueva.

4.1.1.3. Evento *Deactivated*

Cuando se interrumpe la aplicación, por alguna de las razones nombradas en párrafos anteriores, el evento *Deactivated* es lanzado.

Durante la ejecución de este evento se recomienda guardar información importante a nivel de aplicación, ya sea de manera *Transitoria* usando el *PhoneApplicationState* para poder recuperarla al retomar la aplicación luego de la interrupción, o de manera *Persistente* con el *Isolated Storage*, si esta información es independiente de esta instancia de la aplicación.

4.1.1.4. Estado *Dormant*

Luego de desencadenado el evento *Deactivated*, la aplicación entra en estado *Dormant*, esto quiere decir que el sistema operativo suspende la aplicación y a todos sus hilos, manteniéndola en memora. Esto permite que al momento de volver a ella no haya necesidad de recuperar información o crear una nueva instancia de la misma.

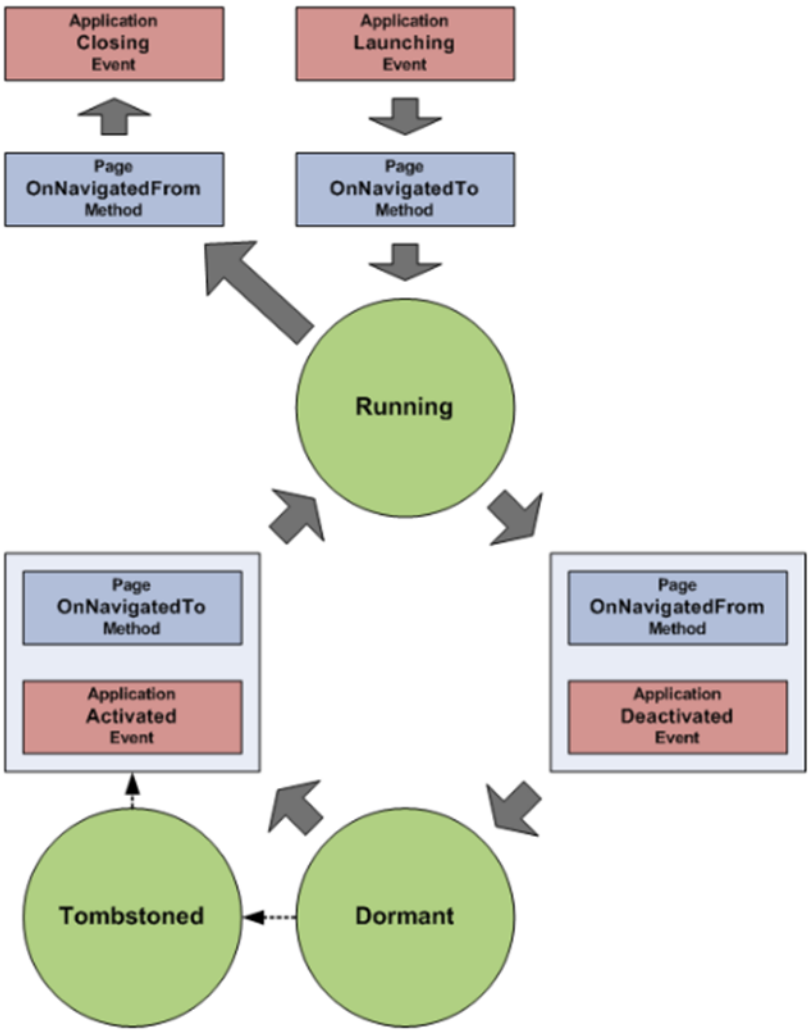

Figura 10 - Ciclo de vida de una aplicación en Windows Phone

4.1.1.5. Estado *Tombstoned*

Si una aplicación entra en estado *Dormant* y se lanzan otras aplicaciones que necesitan memoria para brindar una buena experiencia de usuario, el sistema operativo cambia la aplicación en estado *Dormant* a *Tombstoned*. En este estado la aplicación está básicamente terminada, pero aun al momento de volver a ella permite recuperar información almacenada de manera *Transitoria*.

El sistema operativo solo mantiene hasta cinco aplicaciones en este estado, una vez que sale del mismo la aplicación se encuentra finalizada totalmente.

4.1.1.6. Evento *Activated*

Este evento se dispara cuando el usuario vuelve a la aplicación y esta se encuentra en estado *Dormant* o *Tombstoned*.

Para determinar en qué estado esta, los argumentos de este evento proveen una propiedad llamada *IsApplicationInstancePreserved*. Esta propiedad es de tipo boolean, y si su valor es verdadero entonces la aplicación se encuentra en estado *Dormant* y no es necesario recuperar información alguna, sino que el sistema operativo se encargara de ello. Si el valor de la propiedad el falso, entonces la propiedad está en estado *Tombstoned* y se debe recuperar los datos almacenados de manera *Transitoria*. Si se necesitan recuperar datos almacenados en el *Isolated Storage* o desde la red, se recomienda no hacerlo en este evento sino disparar desde él una *Background Task* que se encargue de realizar esta tarea de manera asincrónica.

4.1.1.7. Evento *Closing*

Este evento se dispara cuando el usuario navega hacia atrás luego de la primera página de la aplicación y produce que la misma finalice. Pero a diferencia de cuando se encuentra en el estado *Tombstoned*, no se preserva información alguna, salvo obviamente la almacenada en el *Isolated Storage* de la aplicación.

En este evento se debe aprovechar a guardar la información importante para la aplicación de manera *Persistente*. Pero se debe tener presente que el sistema operativo dispone para este evento un tiempo máximo de ejecución de diez segundos, luego de este tiempo la aplicación finaliza de manera terminante. Se recomienda no ir acumulando información importante y esperar a que este evento se dispare para guardarla, debido a que puede que esta sea demasiada para la duración del evento.

4.1.2.Notificaciones Push

En Windows Phone las aplicaciones pueden disparar distintos tipos de notificaciones al usuario, cada uno con un objetivo y modo de visualización diferente. Las notificaciones en Windows Phone se conocen como *Push Notifications*. Básicamente es un servicio montado en la nube, que se encarga de enviar notificaciones a los teléfonos. Este es un servicio unidireccional, es decir, que solo el servicio de notificaciones puede enviar notificaciones al teléfono, y este último no puede responderlas.

Este esquema de notificaciones tiene como ventaja que no es necesario que la aplicación se esté ejecutando para poder recibir notificaciones, ya que el servicio se comunica directamente con el dispositivo, ayudando así a optimizar el consumo de batería del mismo. Las Notificaciones son de un tamaño pequeño para que no sea pesado el tráfico, y se pueden priorizar para su envío. Daniel Vaughan en (Vaughan, 2012, pág. 581) profundiza sobre los beneficios del esquema propuesto por Microsoft.

El servicio en la nube que se encarga de enviar las notificaciones se llama *Microsoft Push Notification Service (MPNS)* y es el encargado de enviar las notificaciones de las aplicaciones desean enviar notificaciones a nuestro teléfono.

Para explicar cuáles son los componentes y que se debe hacer para que una aplicación sea capaz de enviar notificaciones a los teléfonos, nos basaremos en la descripción que hace Daniel Vaughan en (Vaughan, 2012, pág. 583) sobre el tema.

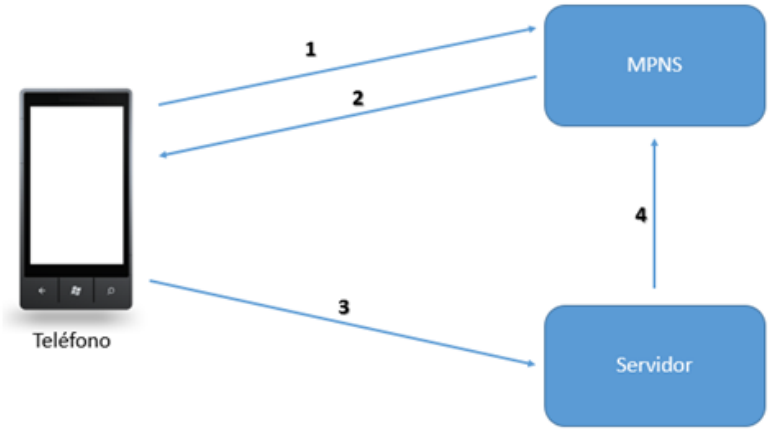

Figura 11 - Modelo de Notificaciones Push

1- La aplicación en el teléfono se conecta con el MPNS para suscribirse al servicio.
2- El MPNS le devuelve una URI que identifica al teléfono y que cualquier servidor puede utilizar para enviarle notificaciones al dispositivo.
3- La aplicación envía esta URI al Servidor de la aplicación que se encargará de elaborar las notificaciones que deben ser enviadas al teléfono.

4- El Servidor envía la notificación junto con la URI del teléfono al MPNS.
5- El MPNS envía la notificación al teléfono.

Como se puede observar, es una implementación sencilla y practica que ofrece una separación de responsabilidades y que permite a los desarrolladores delegar al MPNS la gestión del envió de las notificaciones.

Como se mencionó al comienzo de ésta sección, existen distintos tipos de notificaciones y cada una con un objetivo y modo de visualización. Los tipos de notificaciones son *Tile*, *Toast*, y *Raw*, y para describirlas se tomará de base el articulo *Push Notifications Overview*[11].

4.1.2.1. Notificaciones Tile

Esta notificación actúa sobre el Tile de la aplicación en la pantalla de inicio. Los Tile tienen dos caras, y en cada una de ellas diferente propiedades que pueden ser actualizadas por la notificación.

El frente del Tile cuenta con:

- **Count**: es un contador que va de 1 a 99, cuando es 0 no se visualiza.
- **Background**: es una imagen que actúa como fondo del Tile.
- **Title**: es el título de la aplicación, tiene como máximo 15 caracteres y luego es truncado.

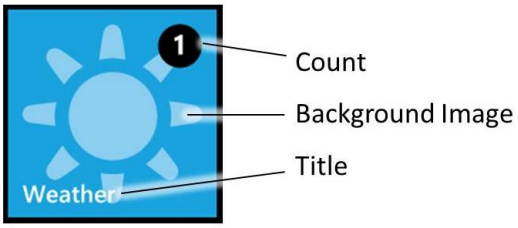

Figura 12 - Frente de un Tile

En la parte de atrás de un Tile se pueden actualizar las propiedades:

- **Back Content**: es una cadena de texto de hasta 40 caracteres.
- **Back Background Image**: es una imagen que aparece en la parte de atrás del Tile.

[11] *Push Notifications Overview* http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402558(v=vs.92).aspx. *Fecha de acceso: 18/03/2013*

- **Back Title**: es el título de la parte de atrás del Tile, tiene la misma longitud que la propiedad Title.

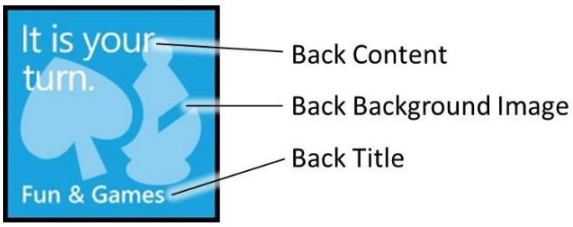

Figura 13 - Parte de atrás de un Tile

4.1.2.2. Notificaciones Toast

Las notificaciones Toast sirven para notificar eventos o información importante, y para hacerlo es visualizada en la parte superior del teléfono ocupando todo el ancho de la misma. Esta notificación se muestra sin importar que tarea este realizando el usuario, y permanece en pantalla 10 segundos. Si el usuario desea dejar de verla debe simplemente arrastrarla hacia la derecha.

Al seleccionarla, tocando la notificación y soltándola, se lanza la aplicación a la que pertenece la notificación. Posee el siguiente conjunto de propiedades:

- **Title**: es el título, se visualiza como texto en *negrita.*
- **Content**: se usa para agregar texto de contenido a la notificación.
- **Parameters**: es un diccionario que puede ser usado para configurar a que página de la aplicación se desea ir al momento de que se inicia cuando se accede a través de la notificación.

Figura 14 - Notificación Toast

4.1.2.3. Notificaciones Raw

Este tipo de notificación es particular, ya que a diferencia de las anteriores, la aplicación debe estar ejecutándose para poder recibirla. Al hacerlo se dispara un evento y el contenido de la notificación es una cadena de información que debe ser interpretada.

4.1.3. Multithreading

Como se describió anteriormente, en Windows Phone solo se permite ejecutar una aplicación a la vez. Sin embargo la necesidad de ejecutar varias tareas al mismo tiempo está contemplada por el sistema operativo.

Windows Phone implementa la ejecución de múltiples tareas de manera simultáneas mediante *Background Task.* Como su nombre lo indica, son tareas que se ejecutan en segundo plano y que no deberían afectar la experiencia de usuario.

4.1.3.1. Background Tasks

Las *Background Tasks* se refieren a la agrupación de un conjunto de acciones modeladas para distintos escenarios, cada una permite realizar en segundo plano la ejecución de acciones determinadas. Las acciones que se pueden realizar son:

- Scheduled Notifications
- Scheduled Tasks
- Background file transfers
- Background audio

De este grupo de acciones solo se entrará en detalle sobre *Scheduled Notifications* y *Scheduled Tasks*, que son las que se usarán en el TFC.

4.1.3.1.1. Scheduled Notifications

El sistema operativo nos permite registrar notificaciones para ser disparadas en un momento indicado, que avisarán de un evento importante o de un simple recordatorio. Estas son las *Scheduled Notifications*, notificaciones que tienen una interface gráfica determinada por el sistema operativo y que cuentan con:

- Titulo
- Texto descriptivo

- Botón para cerrar la notificación
- Botón para posponerla
- Capacidad de abrir la aplicación que registró la notificación al tocar la misma.

Las notificaciones no necesitan que la aplicación se esté ejecutando para ser disparadas, pero son registradas por una aplicación.

Existen dos tipos de notificaciones, Alarmas y Recordatorios. Y ambas se registran mediante el servicio *ScheduledActionService* que proporciona el sistema operativo.

Existen solo dos diferencias entre estas notificaciones:

- Las alarmas siempre tienen el título “Alarma”, mientras que a los recordatorios se les puede cambiar el título que se visualiza.
- A los Recordatorios se les puede asignar una página determinada de la aplicación, y al tocar el recordatorio se lanzará la aplicación en esa página.

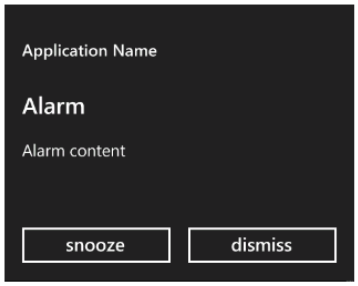

Figura 16 - Alarma UI

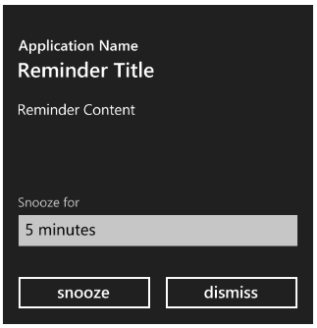

Figura 15 - Recordatorio UI

4.1.3.1.2. Scheduled Tasks

Las *Scheduled Tasks*, como la traducción lo indica, son tareas programables. Esto quiere decir que podemos registrar tareas a para realizar consultar datos de la nube o actualizar datos de la aplicación sin necesidad de que la aplicación se esté ejecutando. Estas tareas tienen un esquema similar al de las *Scheduled Notifications*, se registran mediante el *ScheduledActionService* y las hay de dos tipos, *Periodic* y *ResourceIntensive*. A diferencia de las *Scheduled Notifications* que eran gestionadas por el sistema operativo, estas tareas necesitan de un Agente que las ejecutará. Los agentes se crean a partir de una clase abstracta que proporciona el sistema operativo, llamada *ScheduleTaskAgent*. Es el agente el que contiene la lógica a realizar

cuando se debe ejecutar una tarea, estas últimas sirven para indicar cuando se deben de disparar y de qué tipo son.

El proceso es el siguiente:

1- Se registra la *Task* mediante el *ScheduledActionService*.
2- Cuando el sistema operativo tiene que ejecutar la tarea, llama al *ScheduleTaskAgent* que le corresponde a la tarea.
3- El *ScheduledTaskAgent* realiza su labor.
4- El *ScheduledTaskAgent* informa al sistema operativo el estado en que termino la acción.

Como el Agente formará parte de nuestra aplicación, nos permite consumir datos de la misma. Pero como se dijo anteriormente, no es necesario que la aplicación se esté ejecutando ya que el Agente se dispara en sobre un hilo de procesamiento separado.

Se recomienda registrar las *Task* constantemente, porque se mantienen como máximo catorce días. Es una buena práctica realizar esta tarea cuando la aplicación inicia, y se debe tener en cuenta que no se puede registrar sobre un mismo agente más de una *Task* de cada tipo.

4.2. Desarrollo Ágil de Software

En esta sección se definen los conceptos que ayudaran al desarrollo de la solución de software que se busca conseguir en este Trabajo Final de Carrera.

4.2.1.El Manifiesto Ágil

El Desarrollo Ágil de Software es una corriente en el ambiente de la construcción de software que tiene sus fundamentos en el Manifestó Ágil (Manifesto, *Agile Manifesto*, 2012):

*"**Individuos e interacciones** sobre procesos y herramientas*

***Software funcionando** sobre documentación extensiva*

***Colaboración con el cliente** sobre negociación contractual*

***Respuesta ante el cambio** sobre seguir un plan*

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda."

Además en el manifestó se enuncian los doce principios del software ágil (Manifesto, Principios del Manifiesto Ágil, 2012):

> *"1 - Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continúa de software con valor.*
>
> *2 - Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.*
>
> *3 - Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
>
> *4 - Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.*
>
> *5 - Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.*
>
> *6 - El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
>
> *7 - El software funcionando es la medida principal de progreso.*
>
> *8 - Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.*
>
> *9 - La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.*
>
> *10 - La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
>
> *11 - Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.*

12 - A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia."

Estos principios son el resultado de las observaciones y experiencia de un grupo de expertos, que al observar como las metodologías clásicas no daban resultado y llevaban al fracaso a grandes proyectos de software, decidieron elaborar una alternativa a las formas convencionales en la gestión de proyectos y desarrollo de software.

4.2.2.Iterativo e Incremental

El desarrollo ágil de software propone realizar un ciclo iterativo e incremental. El principio uno, dos, y tres enuncian que en iteraciones de tiempo cortas se buscar responder a los cambios que puedan surgir para darle ventaja competitiva al cliente, entregar software funcionando, y que a medida que pasen las iteraciones se encontrará una versión más madura del producto que se construye.

Lo importante es mantener una serie de iteraciones de un periodo de tiempo no demasiado largo. Durante TFC, el proceso de desarrollo utilizado adoptará esta práctica ágil e implementará iteraciones de no más de una semana, como se verá en la sección 4.3 Metodología Utilizada.

4.2.3.User Stories

Las *User Stories* o historias de usuario son una forma de representar las necesidades de a quién va dirigido el software. Mike Cohn en (Cohn, 2004) las define como:

"Una historia de usuario describe la funcionalidad que será valiosa para un usuario o comprador de un sistema o software. Las historias de usuario se componen de tres aspectos:

- *Una descripción escrita de la historia utilizada para la planificación y como un recordatorio.*
- *Conversaciones acerca de la historia que sirven para dar cuerpo a los detalles de la historia.*
- *Pruebas que transmitir y detalles del documento que puede utilizarse para determinar cuándo una historia completa."*

Las historias de usuario tienen distintos tamaños, a las grandes de las llama *epics* o épicas y las más pequeñas se llaman *smallers* o pequeñas. Generalmente las historias épicas se terminan dividiendo en un conjunto de historias más pequeñas luego de un proceso iterativo.

Las historias de usuario serian la contra parte de los clásicos casos de uso, que en vez de enfocarse en definir en papel todo el flujo de la funcionalidad a desarrollar, son escritas en lenguaje del negocio, priorizando el contexto y la funcionalidad sin adentrar mucho en los detalles de la mismas. Estos detalles se van definiendo a partir de conversaciones con el cliente, y terminan siendo expresados en las pruebas de aceptación.

Se definen con una prioridad y un tamaño, siendo el tamaño una unidad de medida del equipo llamada *Story Points* o Puntos de historia. Los puntos de historia no tienen un valor estándar, sino que cada equipo le da su propio valor. Este puede ser un día ideal de trabajo, o una semana ideal de trabajo, o un cálculo complejo que involucre el tamaño de la historia, el esfuerzo para completarla, y una unidad de tiempo.

La prioridad de una historia de usuario la da el cliente, y en base a ella es que el equipo abordará las historias más prioritarias para entregar al cliente software funcional que evacue sus necesidades más inmediatas luego de cada iteración.

4.2.4. Test Driven Development

Es una técnica para el desarrollo de software y se basa en escribir en forma de test de unidad lo que se quiere construir, y empezar a construir solamente lo necesario para que el test tenga un resultado exitoso.

Carlos Blé Jurado en (Jurado, 2010) brinda una excelente fuente de conocimiento para la utilización de esta práctica y analiza fuertemente los beneficios que proveen al desarrollo de software. El autor en (Jurado, 2010, pág. 53) define a **TDD** como:

> *"**TDD** es una técnica para diseñar software que se centra en tres pilares fundamentales:*
>
> - *La implementación de las funciones justas que el cliente necesita y no más.*

- *La minimización del número de defectos que llegan al software en fase de producción.*
- *La producción de software modular, altamente reutilizable y preparado para el cambio".*

El algoritmo de TDD es simple, pero generalmente la complejidad radica en incorporar el proceso y asumir el cambio en la manera de construir software. El siguiente diagrama de estado explica el algoritmo:

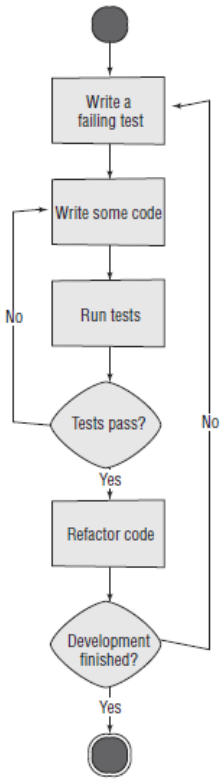

Figura 17 - Algoritmo de TDD (Ibrahim, 2009)

Así como lo explica el diagrama, el proceso de desarrollo consiste en escribir un test de unidad que busque probar una funcionalidad puntual del sistema, pero si haber creado nada aun de ella, es decir, escribir las clases y métodos que se utilizarán para cumplir la funcionalidad y que el test de un resultado positivo pero sin haber creado aun ninguno de los componentes necesarios

para que esto suceda. Obviamente el test no pasará, porque no se podrá compilar el proyecto. Entonces se crean las clases y métodos necesarios para que el proyecto pueda compilar. Una vez hecho esto, se puede ejecutar el test, que va a fallar porque todavía no se ha implementado la funcionalidad. Entonces debemos escribir más código pero solo lo necesario para que el test de verde. Una vez que el test tiene un resultado positivo, refactorizamos lo que acabamos de hacer para eliminar redundancias y código duplicado.

El proceso de refactorización es explicado de manera simple por Martin Fowler en (Fowler Martin, 1999),

> *"Refactorización es el proceso de cambio de un sistema de software de tal manera que no altera el comportamiento externo del código pero mejora su estructura interna. Es una manera disciplinada para limpiar el código que minimiza las posibilidades de introducir errores. En esencia, cuando se refactorizar está mejorando el diseño del código después de que se ha escrito."*

El proceso continua escribiendo otro test de unidad que pruebe otro comportamiento de la funcionalidad que acabamos de probar, por ejemplo, si antes estábamos probando adicionar un producto a un carro de compras, ahora podemos probar que al quitar el producto el carro de compras quede vació. Esto hará que agreguemos o cambiemos funcionalidad de las clases que ya hemos creado, completado su comportamiento. Esto se conoce como técnica de Triangulación, cada test de unidad prueba una funcionalidad puntual, por lo que no podemos debemos agregar más que el código necesario para que cumpla esta funcionalidad, es aquí donde debemos controlar nuestro impulso de querer completar la clase con otros métodos.

La técnica de triangulación nos permite completar las funcionalidades de uno o más objetos del sistema probando diferentes comportamientos de las funcionalidades que se están implementando, y agregando así código siempre utilizable. Para comprobar que todo el código que escribimos se esté utilizando existe una métrica llamada Cobertura de Código, que indica cuanto del código de la aplicación es utilizado por los test de unidad. Y así poder optimizar nuestro desarrollo de software para que no haya porciones de código fuente que no se utilice.

4.3. Metodología Utilizada

Como se adelantó en la sección 3.4 Limitaciones, el equipo de desarrollo de la aplicación que dará como resultado el TFC está compuesto por un único desarrollador y un líder de proyecto. Dado que el equipo es pequeño, en lugar de abordar una metodología se define un proceso de desarrollo de software para el TFC impulsado por el contexto del mismo, al que se definirá como o **S.T.D.Pro** o Small Team Development Process.

4.3.1. Small Team Development Process

Este proceso de desarrollo para equipos pequeños servirá como hilo conductor para la construcción del software que se intenta obtener el TFC.

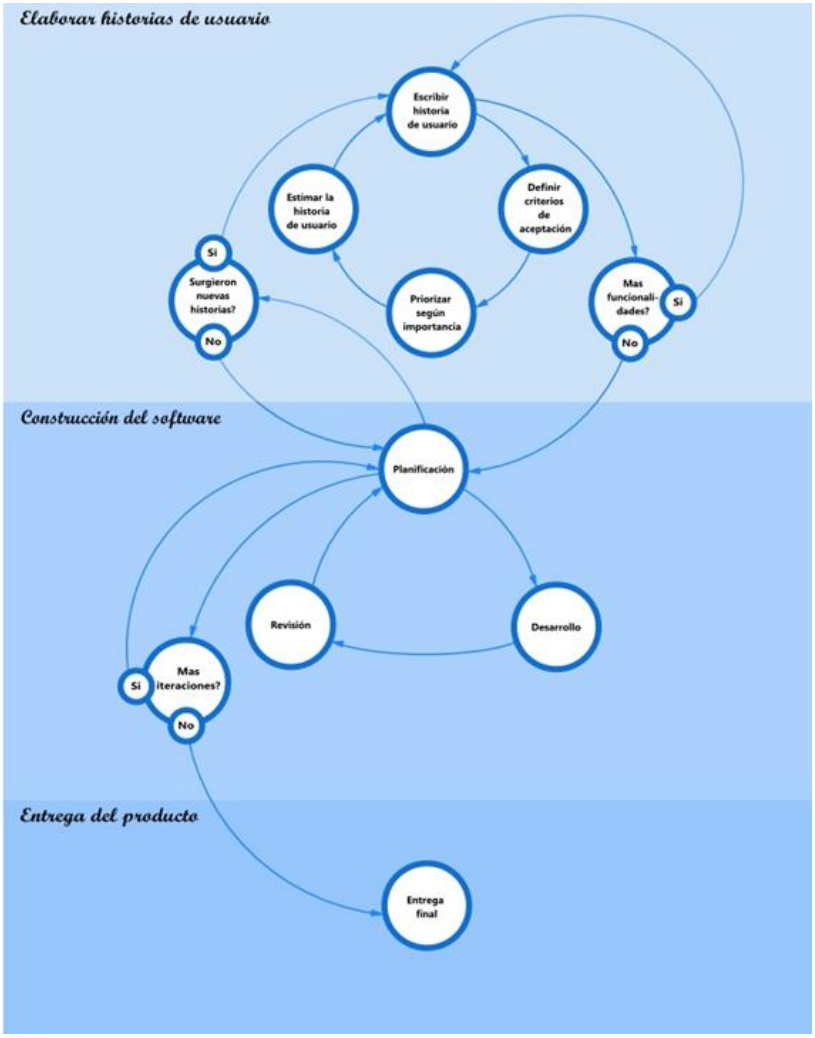

Figura 18 - Small Team Development Process

STDPro, como se puede observar en la Figura 18, está conformado por tres etapas con sus correspondientes pasos, en dos de ellas se realizan interacciones para poder soportar respuestas al cambio y lograr afinar lo que se está buscando, con el único objetivo de que el producto resultante sea el esperado.

4.3.1.1. Elaborar historias de usuario

Como se describió en la sección 4.2.3.User Stories, las historias de usuario sirven para expresar las funcionalidades que debe poseer el software mediante un lenguaje de negocio, y además cuenta con los requisitos necesarios para que la historia este completa. De este modo se busca darle las herramientas necesarias al desarrollador para que pueda realizar la historia y evitar que se produzcan malas interpretaciones.

Durante esta etapa de ***STDPro*** se desarrollaran las historias para la aplicación, y el resultado de la misma es el alcance de la aplicación informática.

Buscando estandarizar el modo de escritura de las historias se tomará como plantilla de historia de usuario la que Craig Brown en (Brown, 2012) comparte. Solo se modificaran los textos en inglés, para que sea más didáctico su uso. Se toma esta plantilla porque expone todos los conceptos descriptos durante el Marco Teórico y permite a simple vista identificar cada uno.

Frente de la plantilla:

Historia de usuario	*[Título de la historia de usuario]*	*Prioridad* ____
Como	*[Rol de usuario de la aplicación]*	*Tamaño* ____
Quiero	*[Descripción de la funcionalidad]*	
Entonces	*[Beneficio obtenido]*	

Parte trasera de la plantilla:

Criterios de aceptación	*[Título de la historia de usuario]*
Contexto	*[Descripción del contexto]*
Cuando	*[Evento 1] [Evento 2] [Etc.]*
Resultado	*[Resultado 1] [Resultado 2] [Etc.]*

De esta manera, queda presentado el modo que se utilizará para escribir las Historias de Usuario en STDPro.

Para cada historia se realizarán una serie de pasos que se describen a continuación.

4.3.1.1.1. Escribir la historia de usuario

Como el título lo indica se busca escribir la funcionalidad que el usuario necesita, en el caso del presente TFC este rol de usuario lo ocupará la Tutora, debido a que las aplicación móvil va dirigida a los alumnos, y ella tiene sobrada experiencia sobre el ambiente académico y cuáles son las principales herramientas que se les deben de brindar.

Como el frente de la plantilla lo indica, la historia debe tener:

- Un título breve que a simple vista informe que funcionalidad se va a describir.

- El Rol que identifica esa funcionalidad, es decir, a qué tipo de usuario le interesa contar con esa funcionalidad.
- Descripción de la funcionalidad.
- El beneficio que obtendría el usuario al poseerla.

4.3.1.1.2. Definir criterios de aceptación

En este paso se escriben distintos escenarios que servirán de ejemplo al desarrollador, sobre lo que se debe obtener al aplicar la funcionalidad que describe la historia de modo que si se cumplen todos los escenarios, la historia estará completa.

En la parte trasera de la plantilla se deben completar:

- Titulo breve de la historia
- Contexto sobre el que se describen los escenarios
- Distintos escenarios que pueden incluir valores a ingresar u otro tipo de dato de entrada
- Resultados esperados para cada uno de los escenarios descriptos

4.3.1.1.3. Priorizar según importancia

La historia debe tener un nivel de importancia, esto se traduce en cuan necesaria es la funcionalidad para el usuario.

Saber su importancia nos permite construir estrategias al armar la iteración en el desarrollo de la aplicación, para que la misma cuente con las funcionalidades más importantes para el usuario.

En el presente Trabajo Final de Carrera se usarán tres niveles de importancia: *Bajo*, *Medio*, y *Alto*.

4.3.1.1.4. Estimar la historia de usuario

La historia de usuario deberá contar con un tamaño, el mismo estará expresado con *Story Points* o Puntos de Historia. Al momento de estimar la historia se debe tener presente que el tamaño de la misma involucra la construcción de todo lo necesario para que la historia se cumpla, pruebe, y entregue al usuario.

Durante el TFC tomaremos como medida inicial que un **Story Point equivale a dos horas diarias de trabajo productivo**.

Esta medida se basa en el tiempo con el que cuenta el autor del TFC para poder construir la aplicación.

4.3.1.2. Construcción del software

Durante esta etapa se realiza la construcción del software, así como el nombre de la etapa lo indica, y el modo de construirlo será de manera iterativa e incremental. Para poder dar fluidez al desarrollo y obtener una rápida retroalimentación, como propone el Manifesto Ágil, las iteraciones duraran una semana como máximo.

Cada iteración dará como resultado una versión de la aplicación, que contendrá la cantidad de historias de usuario que puedan abordarse en ese periodo de tiempo.

Así como en la etapa anterior se contaban con una serie de pasos, esta etapa aborda la construcción del software de la misma manera.

4.3.1.2.1. Planificación

Durante este se evalúa cuáles son las historias que se abordaran durante la iteración, tomando como criterio su importancia.

Este paso lo llevarán a cabo en conjunto la Tutora y el Tesista, cumpliendo la función de exponer sus necesidades y de desarrollador de la aplicación respectivamente.

4.3.1.2.2. Desarrollo

Este paso se refiere a la construcción del software a nivel programático, es decir, escribir código fuente que genere la funcionalidad esperada por las historias que fueron incluidas en la Planificación. Para esto último utilizaremos 4.2.4 Test Driven Development, el escribir los test antes de realizar el código fuente nos ayudará a tener siempre presente los criterios de aceptación de las historias que se proponen completar durante la iteración.

4.3.1.2.3. Revisión

La Revisión se refiere a que luego de cumplirse el tiempo de la iteración, se evalúa respecto de lo planificado que se pudo cumplir y que no. Este es un paso importante para poder mejorar en la próxima iteración, y en lo posible tomar medidas correctivas y preventivas. Es una manera de cumplir con el principio doce del manifestó ágil, *“A intervalos regulares el equipo*

reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia".

Además de realizar una evaluación personal por parte del desarrollador (el autor del TFC), el líder de proyecto (la Tutora del TFC) realizará una evaluación de aceptación sobre la versión del software generada durante la iteración, y será la encargada de dar por finalizadas las historias de usuario involucradas en la iteración. Para esta labor se define el siguiente formato:

Historia de Usuario	*Nombre de la historia*
Realizado	*Objetivos alcanzados*
Observaciones	*Cambios a realizar, sugerencias de mejoras, observaciones varias*
Estado de la Historia	*Completa, Rechazada, En Proceso*

Como puede observarse en la Figura 18, se puede volver desde la etapa de *Construcción del Software* a la de *Elaborar historias de usuario.* Durante el paso de Desarrollo se pueden haber identificado funcionalidades que habían sido tenido en cuenta, estas debe ser evaluadas durante la Revisión y si son consideradas de importancia por el Líder de Proyecto se deben de escribir como historias.

4.3.1.3. Entrega del producto

Esta última etapa es una forma de consolidar lo realizado durante las sucesivas iteraciones de las etapas anteriores.

En ella se lleva a cabo la entrega formal del producto de software desarrollado, el que durante cada Revisión fue validado por el líder del proyecto.

4.4. Patrones de Diseño

Para definir Patrones de Diseño, se tomara como base (Erich Gamma, 1994) que explica de manera simple y practica el concepto.

En (Erich Gamma, 1994) se expone que un patrón de diseño está compuesto por cuatro elementos:

1- **El nombre del patrón**, que se utiliza para describir el problema de diseño, la solución, y la consecuencia en una palabra o dos.
2- **El problema**, describe cuando se aplica el patrón, el problema que intenta resolver y su contexto. En ocasiones viene acompañado de una lista de precondiciones que se deben cumplir para su aplicación.
3- **La solución**, describe los elementos de diseños que se deben construir, sus relaciones, responsabilidades, y colaboraciones. La solución no describe una implementación en particular, sino que el patrón actúa de plantilla que puede ser aplicado en distintas situaciones.
4- **Las consecuencias**, son los resultados de aplicar el patrón. Como impacta en la flexibilidad, extensibilidad, o robustez del software, y permite evaluar si es conveniente aplicarlo o no.

4.4.1. Repository

El patrón Repository o Repositorio, sirve de mediador entre la capa de negocio y la capa de datos, facilitando la solo la información que se necesita de esta última, evitando que en la capa de negocio se realice lógica de búsqueda de datos, separando de esta manera las responsabilidades y disminuyendo el acoplamiento entre capas.

En (Fowler, P of EAA Catalog - Repository, 2012) se puede encontrar una explicación más detallada de este patrón y un diagrama de secuencia del mismo para clarificar el concepto.

4.4.2. Model-View-ViewModel

El patrón de diseño Model-View-ViewModel (MVVM) es presentado por Microsoft como una implementación del patrón PM o *Presentation Model* de Martin Fowler para su plataforma *Windows Presentation Fundation* (Smith, 2013).

Básicamente este patrón implementa una arquitectura de tres capas: Model-View-ViewModel (Britch, Cheung, Kinney, & Sharma, 2012, pág. 25).

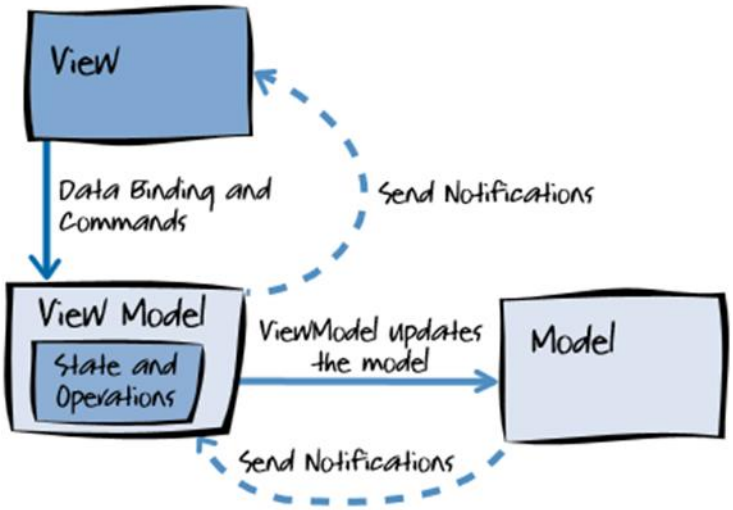

Figura 19 – Representación gráfica del patrón Model-View-ViewModel (Britch, Cheung, Kinney, & Sharma, 2012, pág. 25)

4.4.2.1. View

Es la interfaz gráfica de la aplicación, define la forma y estructura en que el usuario recibe la información. Una Vista puede estar compuesta por múltiples vistas, cada una encargada de proveer información determinada. En la sección 5.2 Construcción del software se verá el uso de las mismas de manera intensiva.

4.4.2.2. ViewModel

El *ViewModel* actúa como intermediario entre las *Views* y el *Model*. Provee información a la *View* mediante el mecanismo de *DataBinding*, el mismo consiste en conectar una propiedad del *ViewModel* al contenido de una sección de la *View*, un bloque de texto por ejemplo. Esta información es a su vez provista por el *Model*, y el *ViewModel* se encarga de realizar transformaciones sobre la misma si fuera necesario, antes de proporcionársela a la *View*.

4.4.2.3. Model

Es el Modelo de datos de la aplicación, puede incluir validaciones de negocio y los objetos de dominio. Es importante no confundir con la capa de datos. El *Model* está compuesto por entidades de negocio.

4.4.2.4. Commands

Una *View* puede contener elementos que disparen eventos, como botones u otros similares. El encargado de resolver estas acciones es el *ViewModel*, pero lo realiza mediante los *Commands*. Los *Commands* se encargan de realizar la acción disparada desde la View, por ejemplo, si se presiona un botón para guardar la información en la *View*, el *command* posee la lógica necesaria para tomar la información del *ViewModel* y enviarla a la capa de datos.

Los *Commands* son parte del *ViewModel*, y se enlazan a los elementos de la *View* utilizando *DataBinding*.

4.4.2.5. MVVM Light

El patrón M-V-VM puede implementarse con facilidad, pero en lugar de eso, se eligió utilizar el framework MVVM Light de Laurent Bugnion[12]. Este framework provee implementaciones de las distintas capas y una plantilla de proyecto que facilita comenzar una aplicación de manera rápida y ordenada. Además es *OpenSource* y muy adoptado por la comunidad de desarrolladores de Windows Phone.

4.4.3.Model-View-Controller

El patrón Model-View-Controller (MVC), o Modelo-Vista-Controlador, es un patrón muy similar al ya descripto M-V-VM en la sección anterior. Propone una arquitectura de tres capas donde la cada una de ellas tiene un propósito definido.

En (Galloway, Haack, Hanselman, Guthrie, & Conery, 2010, pág. 167) se describe este patrón como:

"Model-View-Controller (MVC) es un patrón de arquitectura usado para separar una aplicación en tres aspectos principales:

- ***El Modelo**: es un conjunto de clases que describe los datos con lo que se está trabajando así como también las reglas de negocio de como los datos pueden ser cambiados y manipulados.*

- ***La Vista***: *La interfaz de usuario (UI) de la aplicación.*

[12] http://blog.galasoft.ch/

- ***El Controlador**: Un conjunto de clases que maneja la comunicación del usuario, el flujo general de la aplicación, y la lógica especifica de la aplicación."*

4.4.3.1 ASP.NET MVC

Es la implementación de Microsoft del patrón MVC, para su plataforma Microsoft .NET. Actualmente se encuentra en la versión 4 y es *Open Source*.

4.4.4. Inversión de Control e Inyección de Dependencias

Estos dos patrones, como se explicará a continuación, sirven para desacoplar los componentes de nuestra aplicación y delegar la responsabilidad de la inicialización de esas dependencias.

4.4.4.1. Inyección de Dependencias

La Inyección de Dependencias o *Dependency Injection* (DI), es un patrón que propone dejar bien en claro cuáles son las dependencias de la entidad que se está construyendo y evitar que ella misma las inicialice. En lugar a eso, que sean Inyectadas.

Se pueden Inyectar dependencias en un objeto mediante el constructor o declarando esas dependencias como propiedades.

4.4.4.1.1. Inyección de Dependencias mediante constructor

A continuación se mostrará un ejemplo de la inyección mediante constructor.

```
public class SiaService : ISiaService
{
    private readonly ICareerRepository _careerRepository;
    private readonly ICourseRepository _courseRepository;
    private readonly ICourseCareerRelationshipRepository
_courseCareerRelationshipRepository;
    private readonly IStudentRepository _studentRepository;
    private readonly IStudentCourseRepository _studentCourseRelationshipRepository;

    public SiaService(ICourseRepository courseRepository,
ICourseCareerRelationshipRepository courseCareerRelationshipRepository,
IStudentRepository studentRepository, IStudentCourseRepository
studentCourseRelationshipRepository, ICareerRepository careerRepository)
```

```
    {
        _courseRepository = courseRepository;
        _courseCareerRelationshipRepository = courseCareerRelationshipRepository;
        _studentRepository = studentRepository;
        _studentCourseRelationshipRepository = studentCourseRelationshipRepository;
        _careerRepository = careerRepository;
    }
}
```

4.4.4.1.2. Inyección de Dependencias mediante Propiedad

Este tipo de inyección puede llegar a ser más difícil de entender, porque no deja bien en claro que una clase depende de otra.

```
class Samurai {
  private IWeapon _weapon;
  public IWeapon Weapon {
    get { return _weapon; }
    set { _weapon = value; }
  }

  public void Attack(string target) {
    _weapon.Hit(target);
  }
}
```

4.4.4.2. Inversión de Control

La Inversión de Control o Inversión of Control (IoC), es un patrón que busca implementar una entidad cuya responsabilidad es crear entidades. De esta forma es como ambos patrones, Inversión de Control e Inyección de Dependencias, se complementan. Usando la Inyección de Dependencias se busca dejar expuestas las dependencias y usando la Inversión de Control esas dependencias son inicializadas y entregadas a la entidad que las necesita. De esta manera las responsabilidades están bien definidas y el acoplamiento se mantiene bajo.

4.4.4.2.1. IoC Container

La Inversión de Control, como ya se mencionó, propone delegar la responsabilidad de inicializar entidades a una entidad. Esta última se implementa como un IoC Container, un contenedor donde se define para cada entidad cuál es su dependencia y cómo se construye.

```
public class ServiceModule : NinjectModule
   {
      public override void Load()
      {
         Bind<CloudStorageAccount>().ToMethod(context =>
CloudStorageConfiguration.GetCloudAccount("DataConnectionString"));

         Bind<ITable<Course>>().To<AzureTable<Course>>();
         Bind<ITable<Career>>().To<AzureTable<Career>>();

Bind<ITable<CareerCourseRelationship>>().To<AzureTable<CareerCourseRelationship>>(
);
         Bind<ITable<Student>>().To<AzureTable<Student>>();

Bind<ITable<StudentCourseRelationship>>().To<AzureTable<StudentCourseRelationship>
>();


Bind<IQueue<QueueNotificationMessage>>().To<NotificationQueue<QueueNotificationMe
ssage>>();
         Bind<IQueue<string>>().To<PushNotificationErrorsQueue<string>>();

         Bind<ICareerRepository>().To<CareerRepository>();
         Bind<ICourseRepository>().To<CourseRepository>();

Bind<ICourseCareerRelationshipRepository>().To<CareerCourseRelationshipRepository>()
;
         Bind<IStudentRepository>().To<StudentRepository>();
```

```
            Bind<IStudentCourseRepository>().To<StudentCourseRepository>();
        }
    }
```

Cuando en tiempo de ejecución, no se puede crear una entidad por no conocer la implementación de su dependencia, se consulta al IoC Container y soluciona esta situación.

4.4.4.2.2. Ninject

Un IoC *Container* puede implementarse, pero existen una variedad de *Framework* que ya cumplen esa función. En el caso de este TFC, el elegido es *Ninject*[13].

Ninject es un *framework Open Source* de Inyección de Dependencias para la plataforma .net. Está alojado en *GitHub* y provee una sintaxis simple y fluida para su uso.

La decisión de usar éste *framework* está basada en la experiencia previa que tiene el autor del TFC con este *framework*.

4.4.5. Data Transfer Object

Como se explica en (Fowler, Data Transfer Object, 2013), este patrón consiste en la construcción de un objeto cuya responsabilidad es la de contener información que será enviada desde un sistema a otro, o desde un Web Service a una aplicación para dispositivos móviles por ejemplo.

4.4.6. Reactive Extentions

Reactive extentions[14] es una biblioteca que ayuda a la construcción de aplicaciones de software donde la respuesta a eventos se realiza de manera asincrónica.

Reactive extentions considera una secuencia de datos (movimientos del puntero, solicitudes a una página web, etc.), como colecciones “observables”. De esta manera se suscribe a ellas y cuando se produce un evento que produce un cambio sobre la colección, se realiza de manera asincrónica un proceso definido por el desarrollador.

Es importante destacar que es una librería *open sourse*.

[13] *Ninject http://www.ninject.org, Fecha de acceso: 18/03/2013*
[14] *Reactive extentions http://msdn.microsoft.com/es-es/data/gg577609, Fecha de acceso: 18/03/2013*

4.4.7.JSON

JSON o *JavaScript Object Notation* es un formato de intercambio de información liviano, basado en el lenguaje de programación JavaScript, y que describe estructura de datos de la siguiente forma:

Clase original

```
public class myObject
{
        public string first;
        public string last;
        public int age;
        public string sex;
        public int salary;
        public bool registered;
}
```

Instancia del objeto

```
myObject objeto = new myObject();
objeto.first = “Carlos”;
objeto.last = “Rodrigo”;
objeto.age = 26;
objeto.sex = “M”;
objeto.registered = true;
objeto.salary = 70000;
```

Objeto serializado en formato JSON

```
myObject = {
        "first": "Carlos",
        "last": "Rodrigo",
        "age": 26,
        "sex": "M",
        "salary": 70000,
        "registered": true
```

```
}
```

5. Construcción

En esta sección se realizará el desarrollo de la aplicación académica que busca como resultado del TFC.

Se tomará el proceso de desarrollo **STDPro** definido en la sección 4.3 Metodología Utilizada como hilo conductor para el desarrollo de la misma, construida a partir de las prácticas de Desarrollo ágil de software desarrolladas en el Marco Teórico.

Se realizarán ciclos de desarrollo con un objetivo y al final de los mismos se darán a conocer los resultados del ciclo y se evaluara lo obtenido junto a la Tutora del TFC.

Al finalizar esta sección se podrá contar con la aplicación y obtener comentarios sobre las dificultades que se encontraron durante su desarrollo.

5.1. Elaborar historias de usuario

En la sección 3.3 Alcance del presente documento, se expuso el alcance de la aplicación que se busca construir en este Trabajo Final de Carrera. Pero para cumplir con el proceso de desarrollo adoptado, se deben de escribir las historias de usuario que representan al alcance en el formato establecido.

5.1.1. Historias de usuario

Se presentan las historias de usuario ordenadas por prioridad producto de la primera etapa del proceso STDPro.

5.1.1.1. Prioridad Alta

A continuación se listan las historias de usuario que son consideradas de mayor interés y necesidad.

Historia de usuario	*Listado de materias*	*Prioridad* **Alta**
Como	***Estudiante***	*Tamaño* **20**

Quiero	Poder contar con una lista de las materias de mi plan de estudio.
Entonces	Me permitirá tener un listado de las materias de mi plan de estudio

Criterios de aceptación	*Listado de materias*
Contexto	El plan de estudios de una carrera posee un promedio de 9 materias por año.
Cuando	Se accede a la sección de materias, se visualiza una lista de materias.
Resultado	Se deberá mostrar todas las materias de la carrera mostrando las que el alumno está cursando actualmente y se deberá poder deducir a simple vista el estado de asistencias.

Historia de usuario	*Información de cátedra*	*Prioridad* **Alta**
Como	***Estudiante***	*Tamaño* **10**

Quiero	Poder visualizar información acerca de una cátedra de mi plan de estudio.
Entonces	Me permitirá acceder a la información de la cátedra y sus contenidos

Criterios de aceptación	*Información de cátedra*
Contexto	Una cátedra posee la siguiente información: - Nombre de la materia - Docente de la materia - Día y horario de cursada - Resultado de examen parcial - Resultado de examen recuperatorio - Resultado de examen final - Resultado de trabajos prácticos - Asistencia: Porcentaje actual, Total de horas a dictar, Horas dictadas, Horas asistidas.
Cuando	Se seleccione la materia de la lista de materias
Resultado	Se deberá mostrar toda la información relacionada a la cátedra (ver Contexto).

Historia de usuario	*Aviso para anotarse a examen*	*Prioridad* **Alta**
Como	***Estudiante***	*Tamaño* **5**
Quiero	Recibir un aviso de que debo anotarme a un examen antes de que venza el plazo.	
Entonces	Me serviría de recordatorio y me evitaría perder exámenes	

Criterios de aceptación	*Aviso para anotarse a examen*
Contexto	Cada materia tiene tres fechas de examen (Parcial, Recuperatorio, y Final), y el alumno debe anotarse 72hs hábiles antes de cada una**.**
Cuando	Se debe enviar un recordatorio cuando: • Faltan 96hs para el examen y si el alumno no se ha anotado. • Faltan 84hs para el examen y si el alumno no se ha anotado. • Faltan 73hs para el examen y si el alumno no se ha anotado.
Resultado	Se muestra la notificación con información de la materia, el tipo de examen, y el tiempo restante para anotarse. La notificación debe llevarme a la página de inscripción.

Historia de usuario	*Aviso de cambio de asistencia*	*Prioridad* **Alta**
Como	***Estudiante***	*Tamaño* **5**
Quiero	Recibir una notificación cuando el porcentaje de asistencias cambie en una cátedra	
Entonces	Me permitiría tener presente mi estado respecto de las asistencias en cada cátedra	

Criterios de aceptación	*Aviso de cambio de asistencia*
Contexto	Para la aprobación de las materias, el alumno debe contar con más del 50% de asistencias, en algunas materias más del 75%. También cambian las condiciones de evaluación en exámenes finales si el alumno posee un porcentaje menor al 75%. El porcentaje de asistencia responde a la fórmula: Porcentaje de asistencias = $\frac{Horas\ Dictadas - Horas\ Asistidas}{100}$
Cuando	El porcentaje de asistencias mínimo es del 50% se deben enviar notificaciones si:

	• El porcentaje desciende del 75% • El porcentaje desciende del 50% • El porcentaje asciende del 50% • El porcentaje asciende del 75% Si el porcentaje de asistencias mínimo es del 75% se deben enviar notificaciones si: • El porcentaje desciende del 75% • El porcentaje asciende del 75%
Resultado	Se deben visualizar notificaciones informando la cátedra y que el porcentaje de asistencias ha cambiado.

5.1.1.2. Prioridad Media

Las historias de prioridad media también son importantes pero se evaluó que no necesitan ser abordadas de manera inmediata.

Historia de usuario	*Inscripción a examen final*	*Prioridad* **Media**
Como	***Estudiante***	*Tamaño* **5**
Quiero	Poder inscribirme al examen final de una cátedra	
Entonces	Me permitiría inscribirme al examen desde cualquier lugar.	

Criterios de aceptación	*Inscripción a examen final*
Contexto	Los alumnos deben inscribirse al examen final de una cátedra con 72hs hábiles de anticipación, contando con el porcentaje de asistencia mínimo que exige la cátedra, y habiendo aprobado el examen parcial o el examen recuperatorio del mismo
Cuando	Sea válido inscribirme, al ingresar a una cátedra debo contar con la opción de inscribirme al examen. Si el tiempo de inscripción se terminó, la opción para inscribirme debe desaparecer.
Resultado	Se debe de enviar una confirmación si se pudo inscribir, y no se debe visualizar la opción de inscripción si no se está habilitado para hacerlo.

Historia de usuario	*Acceso a la Aplicación*	*Prioridad* **Media**
Como	***Estudiante***	*Tamaño* **3**
Quiero	Poder acceder a la aplicación utilizando los mismos datos con los que ingreso al SIA	

Entonces	No tendría que generar otras credenciales

Criterios de aceptación	*Acceso a la Aplicación*
Contexto	Los alumnos ingresan al SIA utilizando su Numero de Matricula, Numero de Carrera, y una Contraseña personal
Cuando	Ingreso a la aplicación cargando la credencial de acceso al SIA
Resultado	Logro acceder a la aplicación

5.1.1.3. Prioridad Baja

Las historias que se describen a continuación proporcionan funcionalidad deseada pero no de primordial importancia.

Historia de usuario	*Aviso de publicación de calificación de examen*	*Prioridad* **Baja**
Como	***Estudiante***	*Tamaño* **5**

Quiero	Recibir aviso de que se ha publicado el resultado de un examen y el valor la misma.
Entonces	Me mantendría informado.

Criterios de aceptación	*Aviso de publicación de calificación de examen*
Contexto	Los exámenes poseen una calificación luego de ser evaluados por los docentes.
Cuando	El resultado de un examen es publicado
Resultado	Se debe recibir una notificación con información de que a que cátedra corresponde, el tipo de examen, y la calificación.

5.2. Construcción del software

Contando con las historias de usuario obtenidas de la etapa anterior, es posible comenzar las iteraciones para la construcción del software.

Como ya se describió, el proceso en esta etapa inicia por la Planificación, eligiendo las historias según prioridad. Luego durante el paso de Desarrollo se describirá como se planea abordar técnicamente las historias y desarrollar según lo planeado. Y al finalizar la iteración en el paso de Revisión se evaluará junto a la Tutora del TFC la versión que se pudo obtener en la iteración.

Para poder describir esta etapa de una manera clara y sencilla, cada iteración será descripta como una sección.

Pero antes, se considera oportuno que se describa el conjunto de herramientas utilizadas para el desarrollo del software.

5.2.1. Herramientas utilizadas

A continuación se describen el nombre y la versión de las herramientas que se van a utilizar durante el desarrollo de la aplicación que se busca obtener en el TFC.

- *Sistema Operativo*: Windows 8
- *IDE*: Visual Studio 2012 Profesional
- *Windows Azure SDK v11.0.50727*
- *Windows Phone SDK 8.0*
- *.Net Framework 4.5*

5.2.2.Iteración 1

5.2.2.1. Planificación de la Iteración

Por orden de prioridad, la primera historia a desarrollar ***Listado de materias***. La primera dificultad que se observa es que su tamaño es de **20 SP** y las iteraciones son de una semana. En una semana solamente se pueden realizar **10 SP**, por lo que esta historia de usuario va a ocupar dos iteraciones.

Al ser una historia tan grande, puede ser considerada como una historia Épica. Para este tipo de historia se recomienda que se divida en más historias, pero en este caso esa labor se dificulta porque la funcionalidad a la que apunta la historia es clara y concisa. Para resolver este obstáculo se tomará la decisión de analizar qué es lo que se debe hacer para cumplir la historia y conformar objetivos para cada iteración a partir de lo obtenido en el análisis de la historia.

5.2.2.1.1. Análisis de la Historia *Listado de Materias*

En esta sección se busca identificar las tareas que se deben realizar para que la historia se cumpla.

- Modelar el negocio (solo las materias y la carreras a las que pertenecen) en Windows Azure
- Crear el punto de acceso a la nube para consumir la lista de las materias.
- Crear la aplicación de Windows Phone
- Consumir desde la aplicación la información que está en Windows Azure.
- Guardar esta información en el teléfono móvil.
- Mostrar el listado de las materias en la aplicación.

Al detallar todo lo necesario para cumplir la historia se puede observar con más claridad porque es tan grande la historia.

5.2.2.1.2. Objetivo de la iteración

El objetivo de esa iteración será **construir el modelo de datos** de carreras, materias, y sus relaciones, **crear el punto de acceso en Windows Azure para consumir estos datos**, y por último **crear la aplicación en Windows Phone**.

5.2.2.2. Desarrollo de la Iteración

En base a los objetivos establecidos en la sección anterior, se han modelado los datos de las Materias, Carrera, y su relación. Implementados sobre Windows Azure *Tables*.

En esta iteración se construye el punto de acceso a Windows Azure para consumir la información. Para ello se crea un *Windows* Azure Roles en Windows Azure que implementa un servicio *Windows Comunication Fundation REST*[15] llamado **SiaService**.

Para poder poblar las tablas y administrar la información también se ha implementado un back-end web. Para ello se creó otro *WebRole* de Windows Azure, pero este implementa un sitio web nombrado AdminSite.

5.2.2.2.1. SiaService

Se construyó un *WebRole* de Windows Azure cuya función es servir de punto de acceso para consultar información. En esta iteración se obtiene un servicio con un solo método, *RetriveCoursesByCareer*. Este método se encarga de devolver una lista de materias a partir del código de una carrera. Al implementar el servicio surgió la necesidad de consumir datos, y la forma de hacerlo es por medio de dos *Repositories*, que el servicio recibe en su constructor.

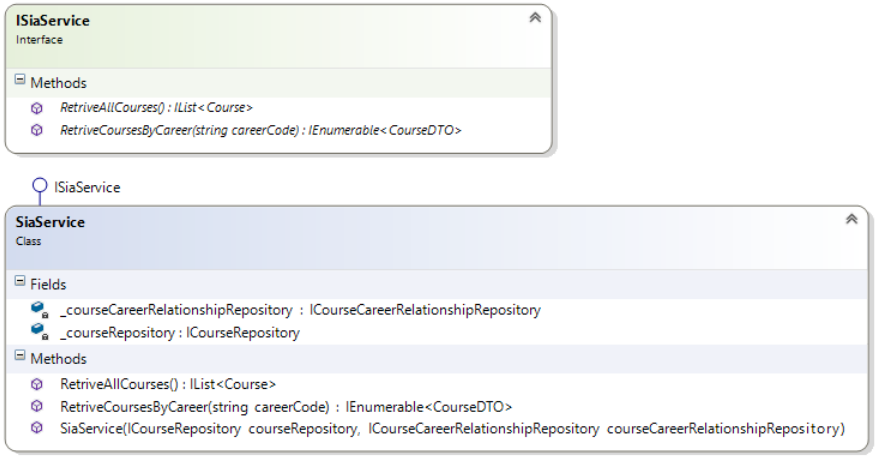

Figura 20 - SiaService

[15] *REST and WCF Services, an Introduction* http://msdn.microsoft.com/en-us/magazine/dd315413.aspx, *Fecha de acceso: 18/03/2013*.

5.2.2.2.1.1 Repositories

La interfaz *ICourseRepository* solo expone un método por el momento, *GetCoursesByCareer*. Y la interfaz *ICourseCareerRelationshipRepository* expone el método *GetCareerYearOfCourseByCareer*.

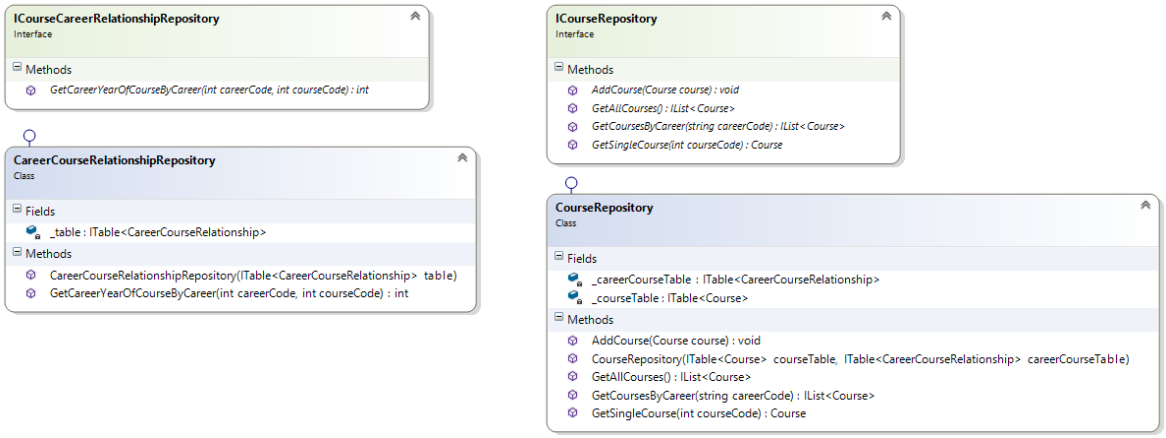

Figura 21 - ICourseRepository y ICareerCourseRelationshipRepository junto con sus implementaciones

5.2.2.2.1.2 Model

El objetivo de estos repositorios es proveer una colección de materias pertenecientes a una carrera y el año en que son dictadas, para construir el objeto *CourseDTO* que es un Data Transfer Object y es la respuesta del servicio.

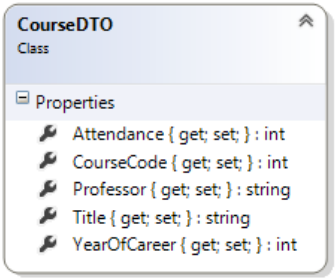

Figura 22 - CourseDTO

El modelo de datos se implementa como *Tables* en Windows Azure, que como se mencionó anteriormente, no responden a un modelo relacional. Sin embargo, permite relacionar

tablas utilizando los dos tipos de claves que poseen, *Partition Key* y *Row Key*, como lo explica Krishnan Sriram en (Krishnan, 2010, p. 281).

Los Repositorios acceden a las Tablas en Azure a través de los objetos AzureTable<T>, que es una implementación de la interfaz ITable<T>, que es genérica[16] pero está restringida a solo a aceptar tipos *TableServiceEntity*, que es una clase provista por el Windows Azure SDK y es una abstracción de la *Table* en Windows Azure.

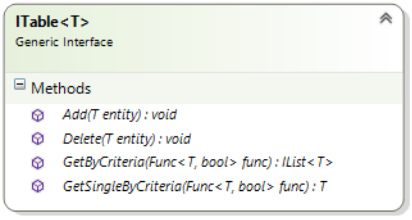

Figura 23 - ITable

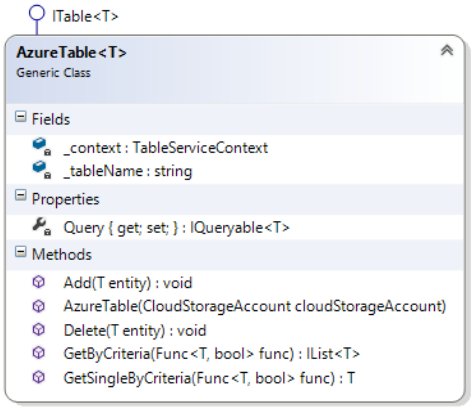

Figura 24 - AzureTable

En el método que expone el servicio SiaService, se obtienen las materias a partir del código de carrera. Esto es un claro signo de una relación entre materias y carrera, sin embargo una cátedra puede pertenecer a más de una carrera, esto es una relación del tipo *many to many,* que se resuelve creando una tabla de relación *CareerCourseRelationship*.

[16] *Generics in .Net Framework http://msdn.microsoft.com/en-us/library/ms172192.aspx. Fecha de acceso: 18/03/2013.*

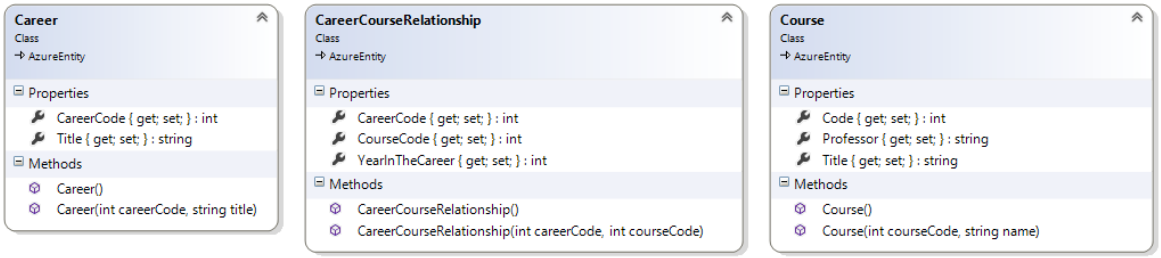

Figura 25 - Esquema Carrera - Cátedra

- **Career:**
  - CareerCode, es la *PartitionKey* de la tabla y representa el código de la misma
  - Title, es la *RowKey* de la tabla y el nombre de la carrera
- **CareerCourseRelationship:**
  - CareerCode, es la *PartitionKey* de la tabla y el código de la carrera
  - CourseCode, es la *RowKey* de la tabla y el código de la cátedra
  - YearInTheCareer, es el año de la carrera donde se dicta la cátedra
- **Course:**
  - CourseCode, es la *PartitionKey* de la tabla y el código de la cátedra
  - Title, es la *RowKey* de la tabla y nombre de la misma
  - Professor, es el nombre del docente que dicta la cátedra

5.2.2.2.2. AdminSite

Este sitio web AdminSite cumple la función de back-end. En él se deben de realizar las tareas de administración de la información académica, tal como dar de alta carreras, materias, o alumnos, asignar alumnos a materias, o materias a carreras. También realizar edición de esta información o eliminar la misma.

Como se mencionó al comienzo de esta iteración, este sitio web esta implementado sobre un *WebRole* de Windows Azure. El mismo se construye utilizando el patrón MVC explicado en la sección 4.4.3 Model-View-Controller.

5.2.2.2.2.1. Model

Para el sitio AdminSite se reutiliza el modelo definido para el servicio SiaService en la sección 5.2.2.2.1.2 Model.

5.2.2.2.2. Controllers

En esta iteración solo se han definido los controllers *CourseController*, *CareerController*, *CareerCourseRelationshipController*, y *HomeController*. En el siguiente diagrama se pueden observar las clases y sus acciones.

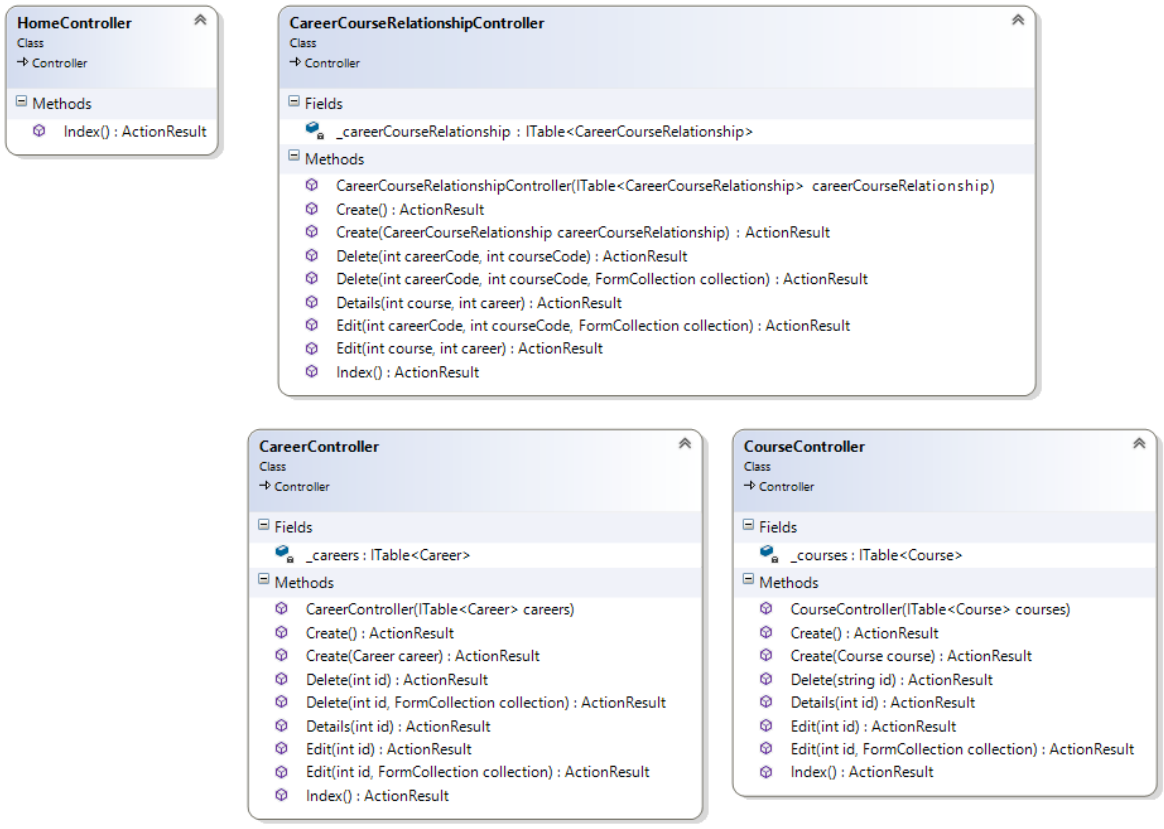

Figura 26 - Controllers generados

5.2.2.2.3. Views

Las vistas para las entidades *Course*, *Career*, y *CourseCareerRelationship*, son simples formularios o listas.

Cursos

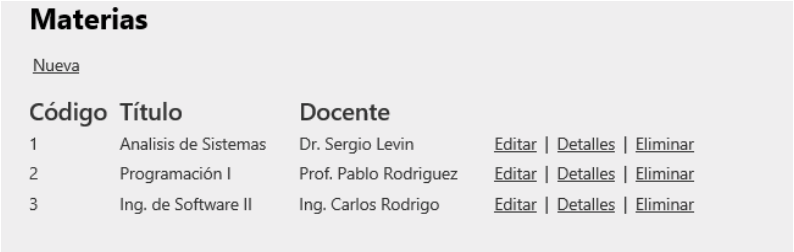

Figura 27 - Listado de Cursos

Figura 28 - Alta y Edición de Materia

Carrera

Figura 29 - Listado de Carreras

Figura 30 - Alta y Edición de Carrera

Relación Carrera-Materia

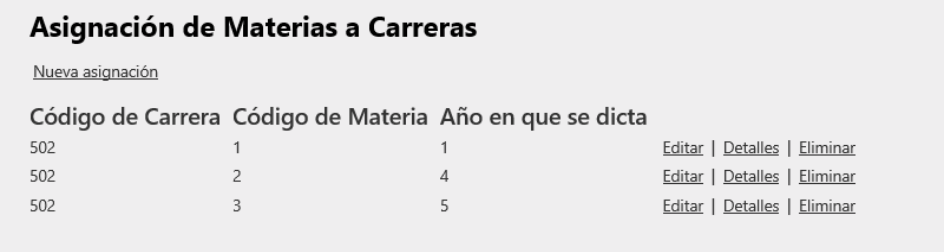

Figura 31 - Listado de Materias asignadas a Carreras

Figura 32 - Alta y Edición de la asignación de una Materia a una Carrera

5.2.2.3. Revisión de la Iteración

Durante el desarrollo no se realizó ninguna mención a la aplicación en Windows Phone, que era uno de los objetivos de la iteración. La razón de esto es porque el desarrollo del sitio de administración AdminSite fue más llevo más tiempo del que se había estimado.

Por lo que la lección aprendida es observar con mayor detenimiento las tareas que involucran las historias de usuario.

De acuerdo con los objetivos iniciales se puede afirmar que el alcance es bueno, solo faltó construir la aplicación para Windows Phone. Esta tarea debe de realizarse en la próxima iteración. Al no haberse completado ninguna historia, el líder de proyecto no tiene la oportunidad de utilizar el formato propuesto para la revisión de la iteración.

5.2.3.Iteración 2

5.2.3.1. Planificación de la Iteración

Durante esta iteración se busca completar las tareas pendientes para dar por finalizada la historia *Listado de Materias*. Y poder contar con una lista de las materias correspondientes a la carrera a la que el alumno está inscripto.

El alcance de la historia es lograr obtener de Windows Azure un listado de materias de una determinada carrera. Como todavía no se implementó la entidad Alumno, se buscará por defecto obtener las materias de la carrera Ing. En Informática. Y por la misma razón, no se podrá visualizar el estado de asistencias de las materias, ya que esta información es relativa al alumno en particular.

5.2.3.1.1. Objetivos de la iteración

Como se describió en la revisión de la Iteración 1, esta iteración tiene como objetivo continuar con las tareas pendientes para dar por cumplida la historia de usuario *Listado de materias*. Las tareas pendientes son:

- Crear la aplicación de Windows Phone que muestre el listado de materias.
- Consumir desde la aplicación la información que está en Windows Azure.
- Guardar esta información en el teléfono móvil.
- Mostrar el listado de las materias en la aplicación.

5.2.3.2. Desarrollo de la Iteración

Las tablas y el servicio construidos en la iteración anterior serán el punto de acceso para que la aplicación en Windows Phone obtenga información contenida en Windows Azure.

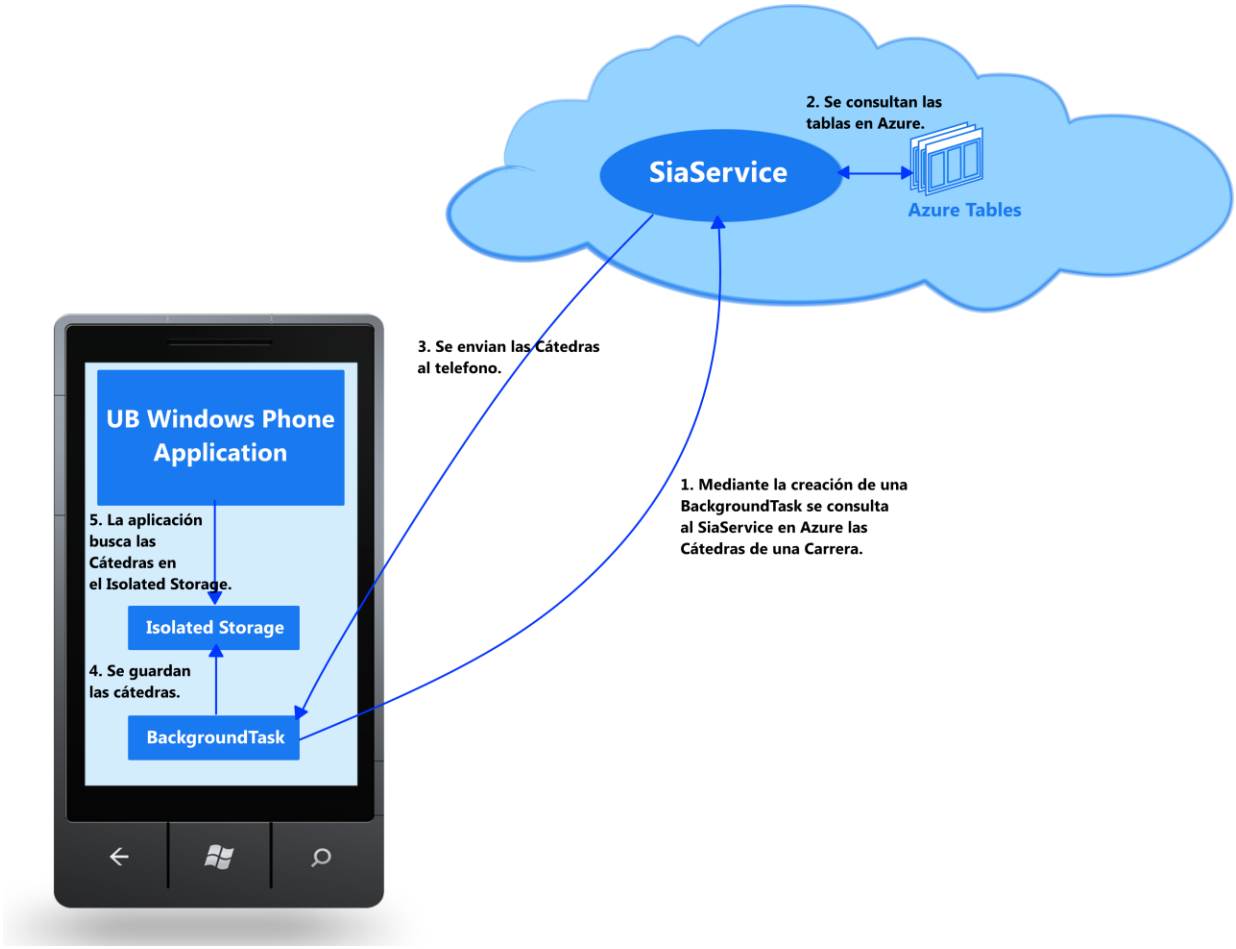

Figura 33 - Estrategia para obtener información de Windows Azure desde Windows Phone

La estrategia descripta en la Figura 33 está inspirada en (Britch, Cheung, Kinney, & Sharma, 2012, pág. 83) y consiste en:

1- Implementar un **ScheduledAgent** que, cuando se dispare una **PeriodicTask**, consuma el servicio **SiaService** creado en la Iteración 1.

2- El servicio consultará los datos de las Azure Tables **Career**, **Course**, y **RelationshipCareerCourse**.
3- El servicio **SiaService** crea una **IEnumerable<CourseDTO>** a partir del resultado de la consulta hecha en 2, y retorna la colección serializada en formato JSON.
4- El **ScheduledAgent** guarda los datos en el **IsolatedStorage** de la aplicación.
5- Al iniciar, la aplicación toma los datos desde el **IsolatedStorage** y los muestra.

En las aplicaciones para dispositivos móviles es de suma importancia que las llamadas a servicios o consumo de datos sean realizados de manera asincrónica. La aplicación no debe de dejar de responder en ningún momento. Es por eso que los paso 1 y 4 son realizados usando *Reactive Extentions*, que nos permite realizar llamadas asincrónicas.

La aplicación Mobile es construida utilizando el patrón M-V-VM, por lo que el resultado de aplicar el mismo da como resultado los siguientes componentes:

5.2.3.2.1 Aplicación Windows Phone

5.2.3.2.1.1 Views

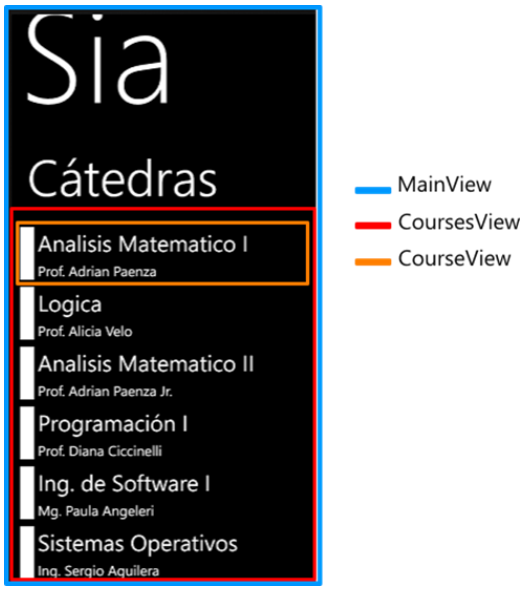

Figura 34 - Views

La Figura 34 muestra la Interfaz Gráfica obtenida durante la presente Iteración. La misma está compuesta de un conjunto de *Vistas* que tienen un objetivo particular. *MainView* es la vista principal de la aplicación, contiene un Control Panorama con un solo Panel denominado *Cátedras.* Dentro de este último está contenida la vista *CoursesView*, cuyo objetivo es mostrar la lista de materias correspondientes. Cada una de ellas se muestra mediante la *CourseView*, donde se puede visualizar el Nombre, Profesor, y estado de asistencia del alumno en la cátedra.

5.2.3.2.1.2 ViewModels

Continuando con la aplicación del patrón MVVM, se obtienen los *ViewModels* para las vistas ya descriptas. Estos heredan de la clase abstracta *ViewModelBase* provista por el framework MVVMLight. También se implementó la clase *ViewModelLocator*, cuyo fin se explicó en la sección 4.4.2 Model-View-ViewModel.

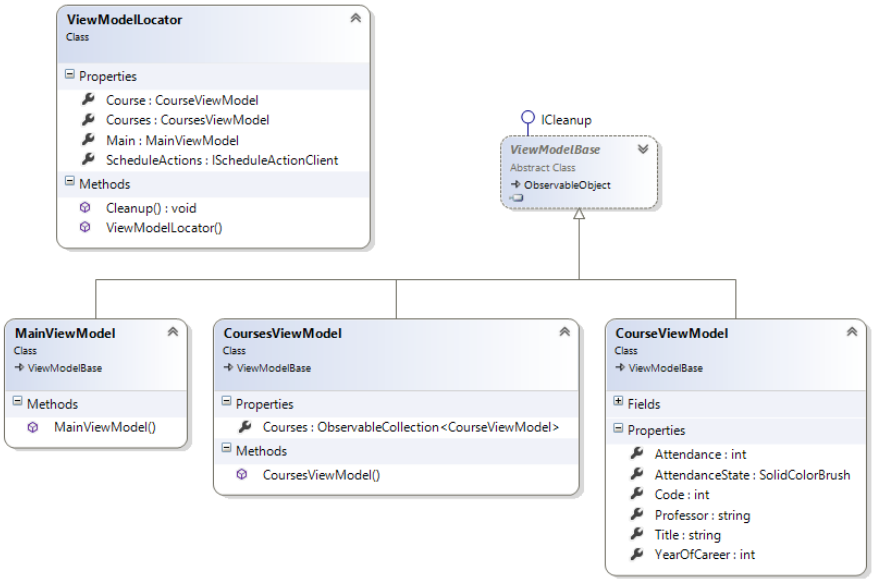

Figura 35 - ViewModels

5.2.3.2.1.3 Model

En cuanto a la tercera parte del patrón MVVM, el Modelo, se implementa la siguiente clase *Course*.

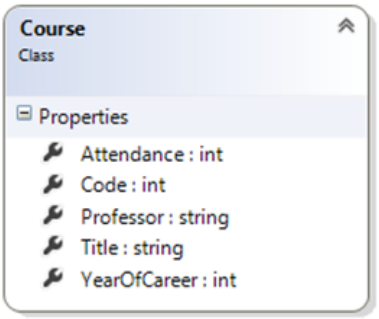

Figura 36 - Course

Para lograr la comunicación entre Windows Azure y Windows Phone también se construyeron.

5.2.3.2.1.4 Repositories

Al igual que en la Iteración 1, se aplica el patrón Repository para consumir los datos del *IsolateStorage* de la aplicación. Este último es descripto en la sección 4.3.2 Ciclo de vida de una aplicación.

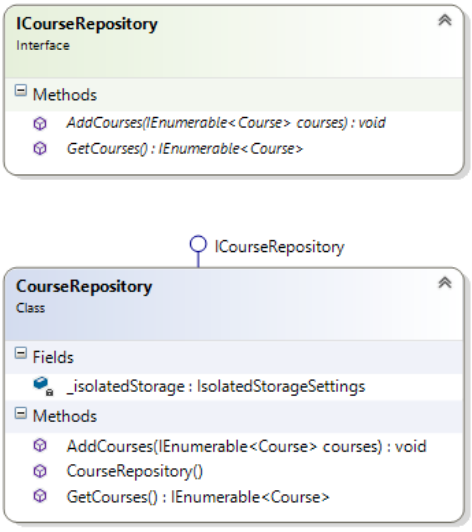

Figura 37 - ICourseRepository y su implementación CourseRepository

5.2.3.2.1.5 Services

Durante la iteración surgió la necesidad de implementar una serie de Servicios con objetivos específicos. *CloudService* encargado de consumir el servicio alojado en Windows Azure, y *SyncronizationService* cuyo objetivo es orquestar el proceso de sincronización de la información acerca de las materias.

Al tratarse de operaciones asincrónicas, el resultado es un *TaskSummary* cuyo estado lo determina el enumerable *TaskResult*.

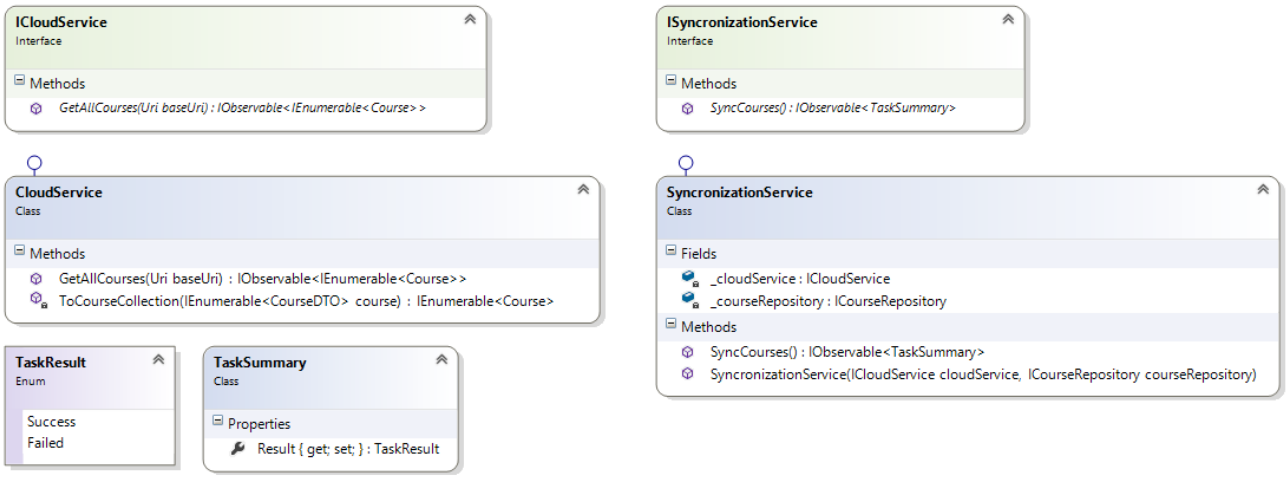

Figura 38 - Services

5.2.3.2.1.6 Agent

El ScheduledAgent de la aplicación móvil tiene la siguiente forma:

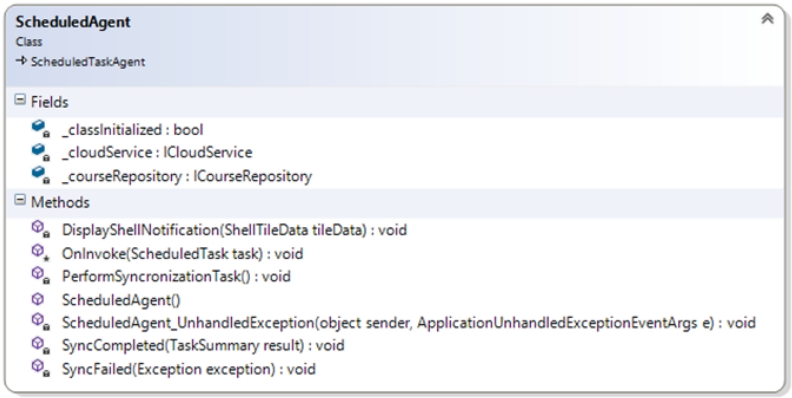

Figura 39 - ScheduledAgent

5.2.3.3. Revisión de la Iteración

De esta iteración se pudo obtener una primera versión de la aplicación para Windows Phone. La misma muestra el listado de materias correspondientes a una carrera, informado el nombre de la cátedra, el docente que la dicta, y por último un indicador de la situación del alumno respecto a las asistencias para con la cátedra. Esta última funcionalidad no está desarrollada hasta el momento, el motivo de ello radica en que como el avance es iterativo, todavía no se ha modelado al alumno. Por lo que el estado de las asistencias no puede ser mostrado todavía.

Con esta primera versión, se cumple la historia de usuario *Listado de Materias*.

Se podría considerar como hito de la iteración, y del proyecto, el haber establecido la comunicación entre las dos plataformas, Windows Phone y Windows Azure. Se lo considera de esta manera, porque durante las historias restantes esta comunicación se usará de manera constante, por lo que el haber logrado implementarla desde la primera historia permite más dinamismo en desarrollo del resto.

El código fuente de esta iteración se encuentra en el Anexo 2 – Iteración 2.

A continuación la tutora del presente TFC realiza su evaluación sobre la iteración:

Historia de Usuario	***Listado de Materias***
Realizado	Realizado en su totalidad
Observaciones	Se recomienda utilizar colores institucionales de la Universidad de Belgrano, como azul, celeste, entre otros.
Estado de la Historia	FINALIZADA, no se detectan errores

5.2.4.Iteración 3

Cumpliendo la primera historia se logró obtener una versión acotada de la aplicación, y establecer la arquitectura general para la misma. En esta iteración se busca seguir agregando valor mediante la inclusión de nuevas funcionalidades.

5.2.4.1. Planificación de la Iteración

La próxima historia es *Información de Cátedra*, la misma consiste en poder navegar en una Cátedra, visualizando su información y la del alumno con relación a la cátedra. Ésta historia está estimada en **10 SP**, y va a ser el objetivo de esta Iteración.

Se considera importante notar como el avance en iteraciones expone como las entidades de negocio y el diseño emergen. En la iteración pasada se pudo ver la necesidad de modelar la entidad Alumno, y en la presente también se necesita información de ella. Sin embargo, no se modelará por completo. Sino que al igual que en la iteración pasada, solo se modelará lo necesario para cumplir con la historia. Esto es, su relación con una cátedra.

El resultado de esta Iteración será una nueva versión de la aplicación, donde además de visualizar el listado de materias correspondiente a la carrera, sino que también se podrá navegar por cada una de ellas obteniendo información de la cátedra y del alumno para con ella.

5.2.4.1.1. Objetivo de la Iteración

En la presente Iteración se busca **comenzar y finalizar la historia de usuario *Información de Cátedra***.

5.2.4.2. Desarrollo de la Iteración

El modo de mantener sincronizada la información será extendiendo la estrategia empleada en la Iteración 2, logrando que no solo se obtenga el listado de materias de forma periódica, sino que también la información correspondiente a cada una de ellas. Se agrega la funcionalidad que al ingresar a cada cátedra también se busque actualizar la información de la misma e impactarla en el *IsolatedStorage* luego de actualizar la vista.

Como todavía no se aborda la historia que involucra el acceso a la aplicación con usuario y contraseña, se continúa trabajando con datos de prueba como en las iteraciones pasadas.

5.2.4.2.1. Actualización de la información de las Materias

Respecto a la implementación de la iteración 2, en la presente se busca obtener y guardar la entidad *Student*. En esta iteración, la entidad *Student* solo contiene una colección de *StudentCourse*, que representa la relación del Alumno para con sus Materias. Una vez obtenida la entidad, se dispone a guardarla en el *IsolatedStorage* junto con cada uno de los *StudentCourse* que contiene. Estos últimos se guardan de manera individual, lo que permite poder actualizarlos de manera independiente.

5.2.4.2.2. Aplicación Windows Phone

5.2.4.2.2.1. Model

Del lado de la aplicación de Windows Phone el modelo está conformado por los DTOs que envía el servicio SiaService, que con la incorporación de la entidad *Student* y *StudentCourse* toma la siguiente forma:

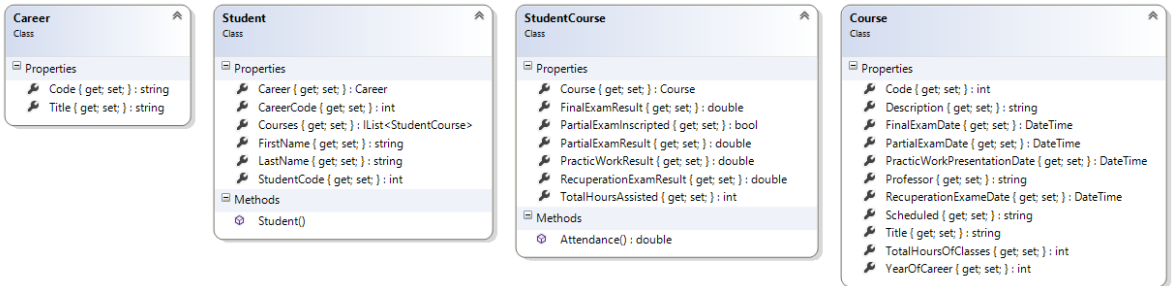

Figura 40 - Model

Las entidades nombradas en el párrafo anterior son descriptas con más profundidad en las próximas secciones.

5.2.4.2.2.2. Views

Durante esta iteración se ha implementado la vista para visualizar la información de una cátedra. Para generar una experiencia de usuario agradable y un buen manejo de la información, se utiliza el Control Pivot, visto en la sección 4.3.1 Silverlight para Windows Phone. El control tiene dos *PivotItems*, **Descripción** y **Mi Situación**, cada uno representado por una vista en particular como se verá en la próxima sección. El flujo de uso de la aplicación termina siendo el siguiente:

Figura 41 - Flujo para la navegación de las Materias

La primera pantalla del flujo es la obtenida durante la Iteración 2, solo que se ha modificado el color del fondo de la aplicación. En la Figura 42 puede observarse el resultado obtenido en la iteración anterior, pero con una serie de mejoras. Cambio en el color de fondo de la aplicación, y lo más destacado, el indicador del estado de asistencias del alumno en una cátedra en particular.

Figura 42 - MainView, CoursesView, CourseResumeView

La segunda pantalla se visualiza luego de seleccionar una cátedra. El propósito de la misma es mostrar la información de la cátedra seleccionada y la situación del alumno respecto de ella. Como se adelantó al comienzo de esta sección, la vista está implementada por *Control Pivot* y posee dos ítems, uno para mostrar la información general de la cátedra y otro para mostrar la situación del alumno. Ambos están implementados como vistas separadas, como puede observarse en las siguientes figuras:

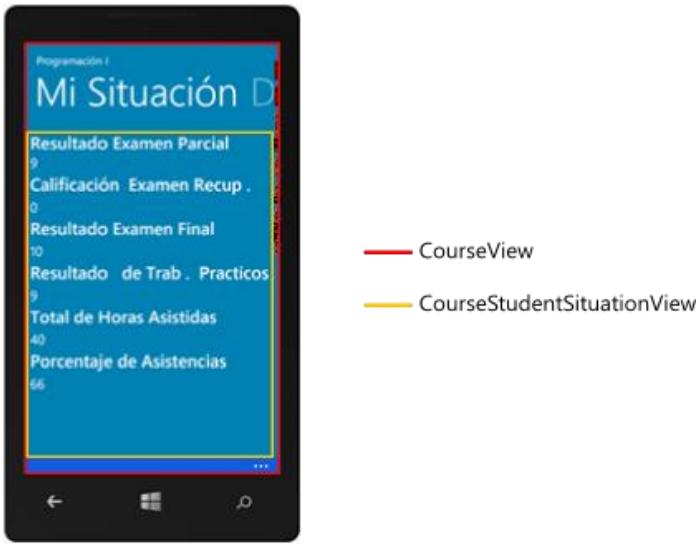

Figura 43 - CourseView, CourseStudentSituationView

La vista *CourseStudentSituationView* muestra la situación del alumno respecto a la cátedra, como ser los resultados de los exámenes, el total de horas asistidas, y el porcentaje de asistencias al momento.

En la Figura 44 se puede observar la vista *CourseDescriptionView*, que permite visualizar la información de una cátedra en particular, mostrando descripción, docente, horarios de la misma, fechas de exámenes, y el total de horas de la cátedra.

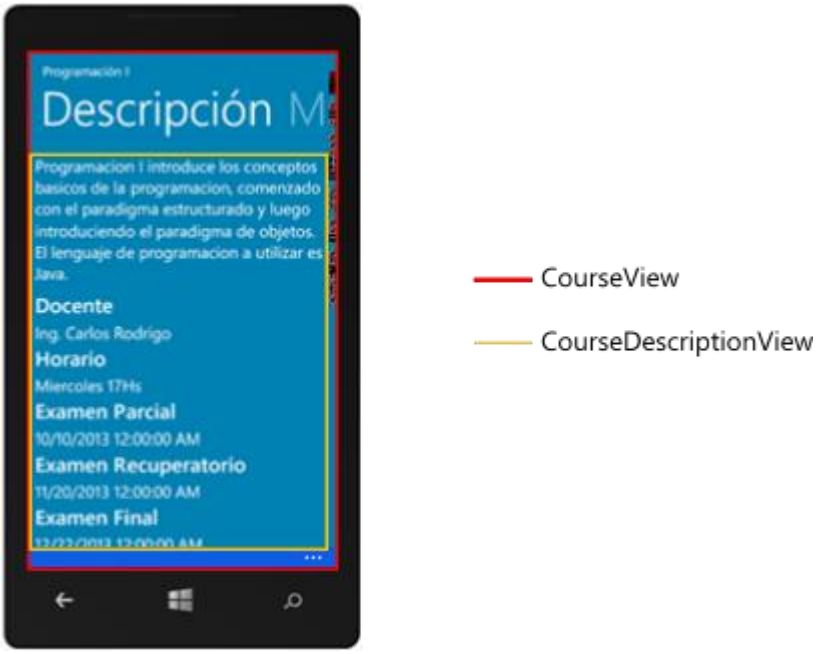

Figura 44 - CourseView, CourseDescriptionView

En las figuras anteriores puede observarse una barra de un color más oscuro con tres puntos blancos al lado derecho. La misma en una barra de aplicación, un componente de Windows Phone que permite agregar acciones e ítems de menú. Es ella donde está alojada la acción de actualizar la información de la cátedra.

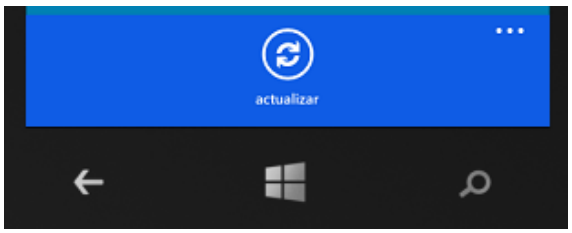

Figura 45 - Acción Actualizar

Al ejecutarse la acción de actualizar, se informa al usuario de estado de la misma mediante una barra de progreso que se despliega en la zona superior de la pantalla.

Figura 46 - Estado de la acción de Actualizar

5.2.4.2.2.3. ViewModels

Al igual que con el nombre de la vista, el ViewModel *CourseViewModel* fue renombrado a *CourseResumeViewModel*. Para dar lugar a que el nombre CourseViewModel lo ocupe una entidad que realmente se identifique con ese nombre. También se crearon los *ViewModels* *CourseDescriptionViewModel* y *CourseStudentSituationViewModel* que proporcionan información a las vistas.

El hito de la iteración es la creación de una clase abstracta *ViewModel* que extiende de la clase *ViewModelBase* provista por *MVVMLight*. El motivo esta implementación es la necesidad de proveer desde los *ViewModels* comandos que se responsabilicen del comportamiento de las paginas en los eventos *OnNavigationTo* y *OnNavigationFrom*.

El esquema resultante se muestra en la Figura 47.

Figura 47 - ViewModels CourseDescriptionViewModel, CourseStudentSituationViewModel, y CourseView

Es oportuno destacar que, el *ViewModel CourseViewModel* contiene a los ViewModels *CourseDescriptionViewModel* y *CourseStudentSituationViewModel*. Esto concuerda con cómo fue construida la Vista *CourseView*, este ViewModel le proporciona toda la información necesaria y organizada conceptualmente.

Puede observarse además, que la entidad *ViewModel* posee una propiedad del tipo *INavigationServices*, y que *CourseViewModel* posee un *ISyncronizationService* y un *IStudentCourseRepository*.

Estas entidades serán desarrolladas en las siguientes secciones.

Con la incorporación de los ViewModels descriptos, el diagrama resultante es el siguiente:

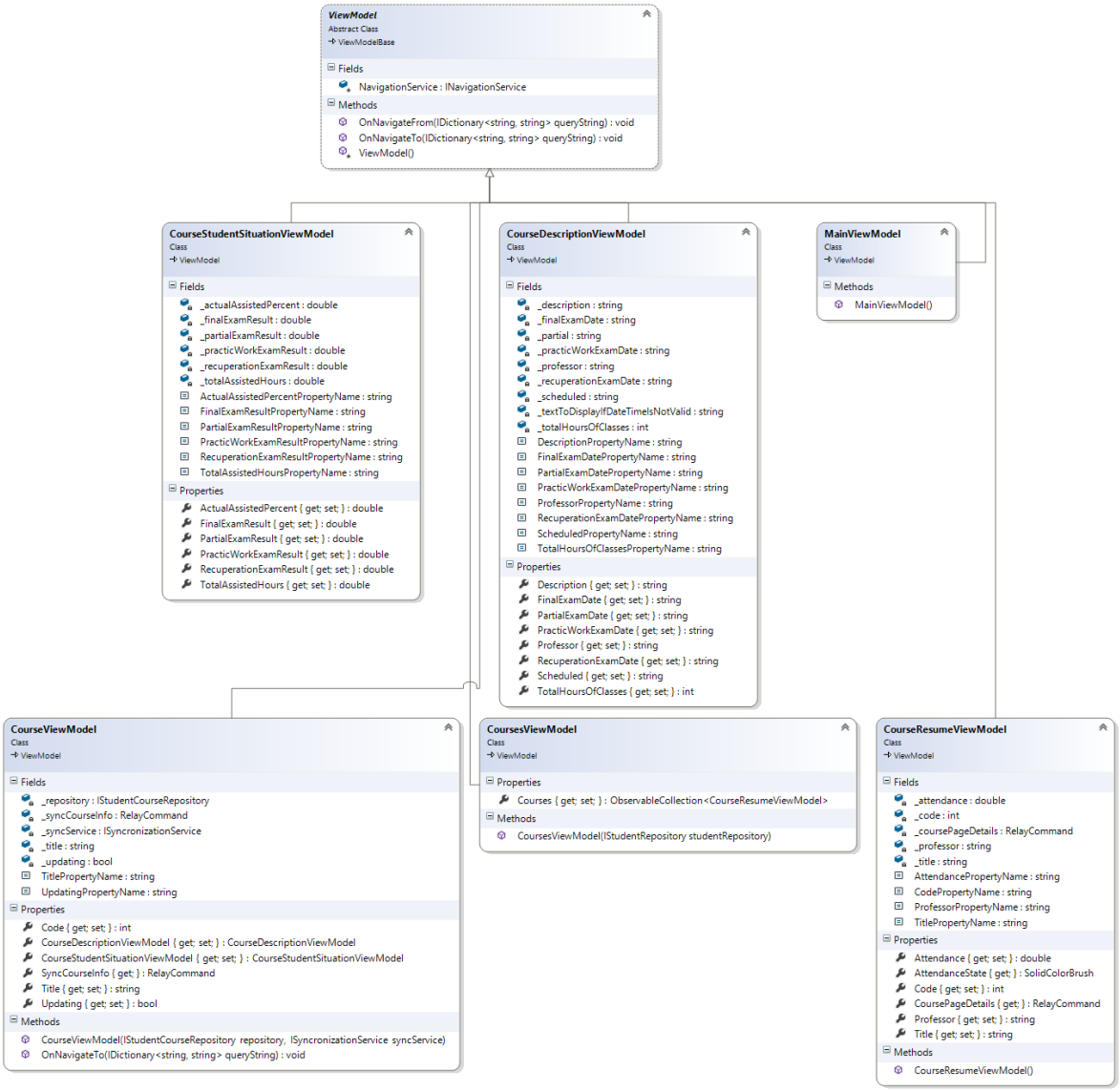

Figura 48 - ViewModels

Como se adelantó en la sección anterior, se ha incorporado la interfaz **INavigationService** y su implementación **NavigationService**. El objetivo de esta es desacoplar el uso del servicio navegación que provee Windows Phone, desarrollado en la sección 4.3.2 Ciclo de vida de una aplicación, y utilizar una implementación de *INavigationService* que lo consuma. De esta manera se puede mantener el bajo acoplamiento y realizar pruebas de unidad más eficientes.

Cada *ViewModel* tiene la responsabilidad de definir si hay una navegación a otra página de la aplicación, pero el servicio *NavigationService* es el encargado de hacerla efectiva.

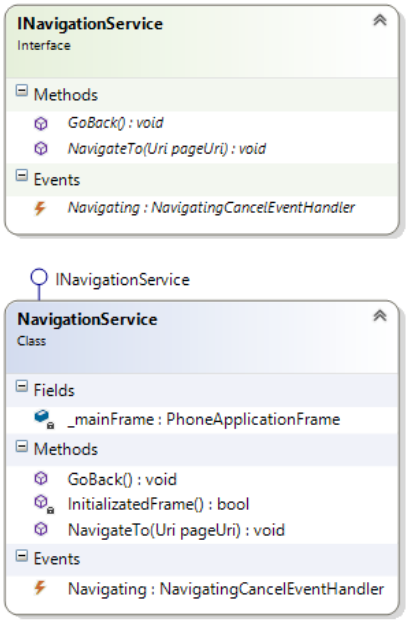

Figura 49 - NavigationService

También se actualizaron los servicios *CloudService* y *SyncronizationService* para que soporten la funcionalidad de actualizar la información de una cátedra en particular.

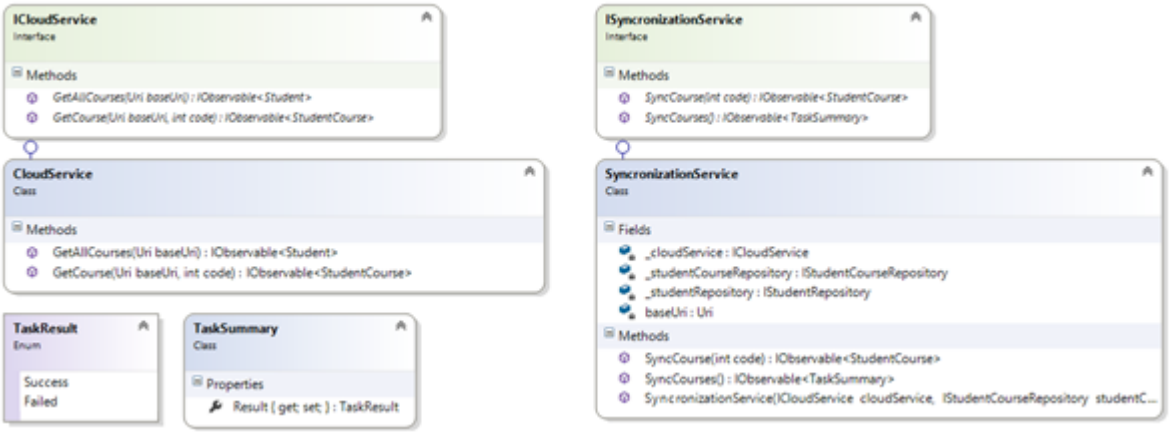

Figura 50 - CloudService y SyncronizationService

5.2.4.2.2.5. Repositories

Respecto a los *Repositories*, se han creado dos nuevos que reemplazan al *CourseRepository* de la Iteración anterior. *StudentRepository* con el objetivo de obtener toda la información del estudiante y guardarla, y *StudentCourseRepository* enfocado solamente a gestionar la relación Cátedra-Alumno.

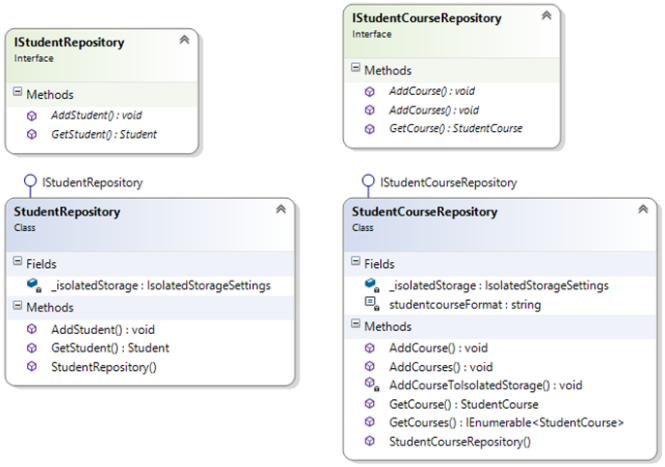

Figura 51 - Repositories actualizados en la Iteración 3

5.2.4.2.3. SiaService

El servicio SiaService también ha sufrido modificaciones, las mismas se pasan a detallar a continuación.

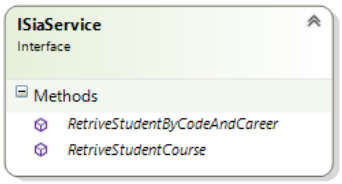

Figura 52 - ISiaService

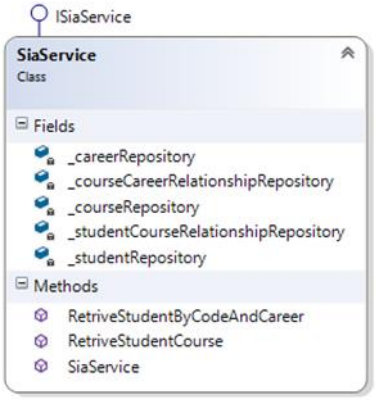

Figura 53 - SiaService

5.2.4.2.3.1. Model

El modelo de se ve afectado por la inclusión de la entidad *Student* y *StudentCourseRelationship.*

StudentCourseRelationship tiene como objetivo representar la relación entre el alumno y una Cátedra, conteniendo información acerca de la situación del alumno respecto de ella, como calificaciones en exámenes, y horas asistidas. Esto último nos permite calcular el porcentaje de asistencias del alumno y lograr completar el indicador de asistencias en la vista *CourseResumeView* (en la iteración 2 tenía el nombre *CourseView*, en la sección Views de la presente iteración se explica el motivo del cambio de su nombre).

Además, la entidad *Course* adquiere datos que se describen como criterio de la historia *Información de materia*, como se puede observar en la figura que se muestra a continuación.

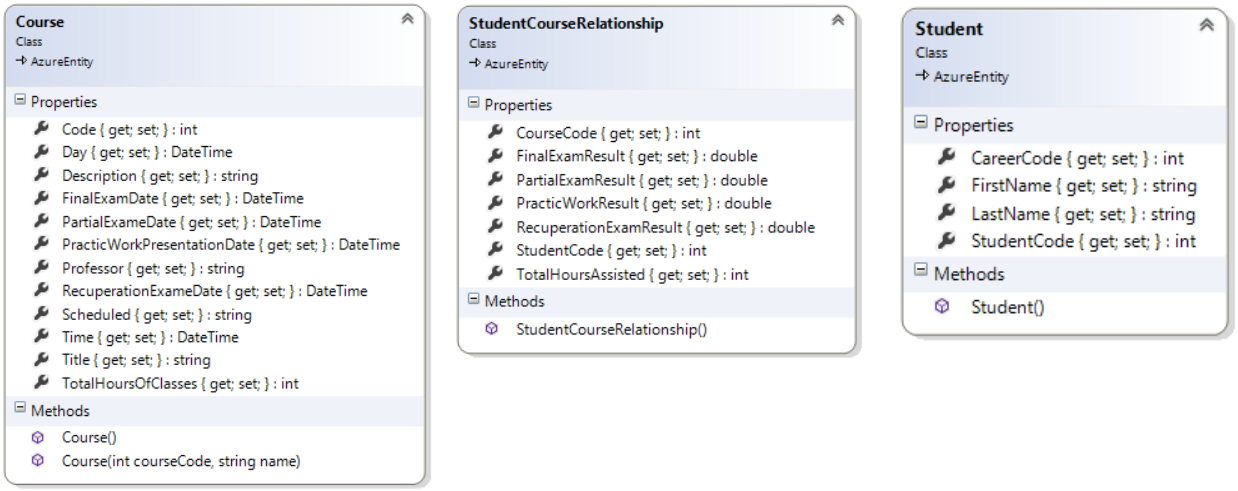

Figura 54 - Relación Student y Course

En la relación *StudentCourseRelationship* la *PartitionKey* es el *StudentCode* y la *RowKey* es el *CourseCode*. Esto se define basándose que la entidad representa a la situación del alumno respecto de una materia, por lo que los datos en Windows Azure deberían agruparse en torno al alumno.

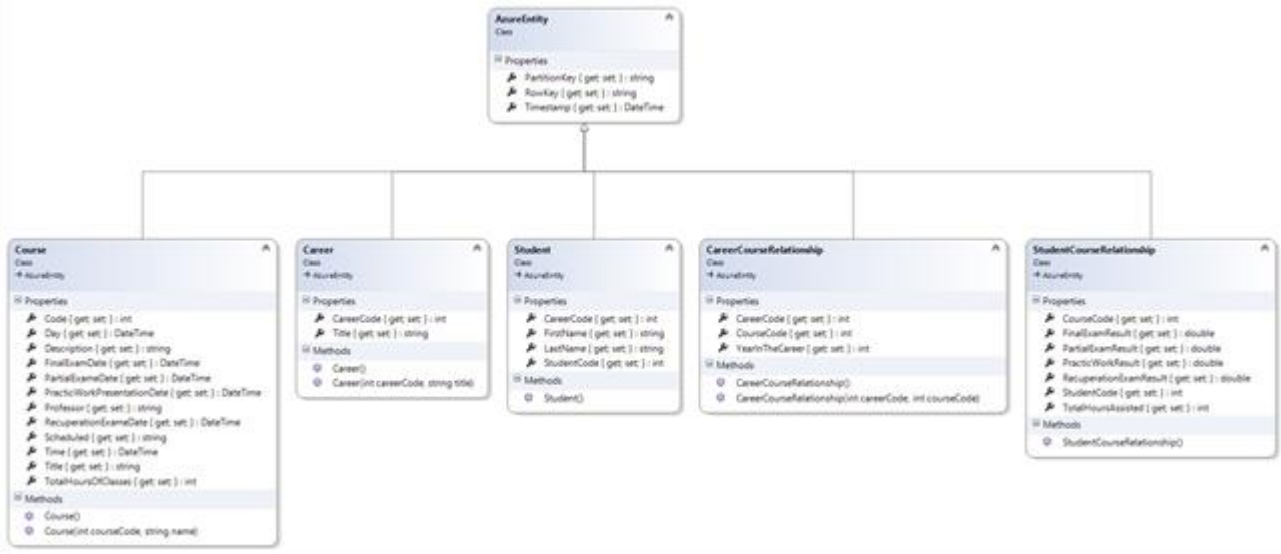

Figura 55 - Modelo de Datos Iteración 3

5.2.4.2.3.2. Repositories

De manera muy similar que en los *Repositories* de la sección anterior (5.2.4.2.2.5 *Repositories*), los repositorios nuevos creados en la presente iteración para el servicio SiaService, están destinados a la manipulación es las entidades *Student* y *StudentCourseRelationship*.

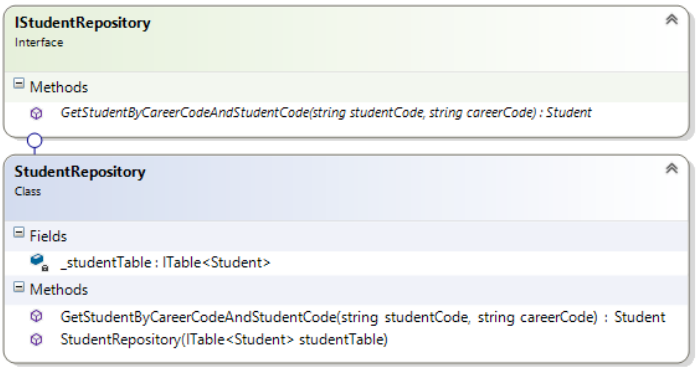

Figura 57 - StudentRepository

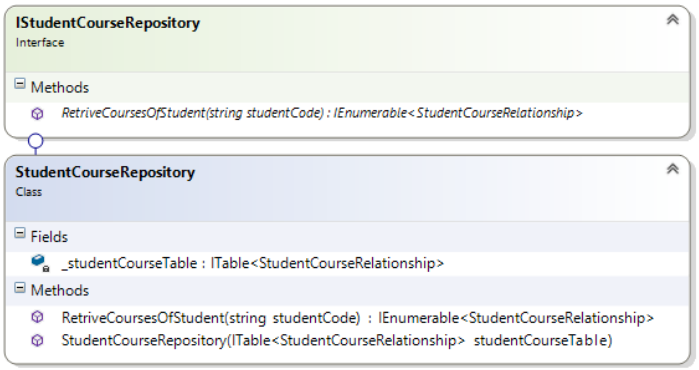

Figura 56 - StudentCourseRepository

5.2.3.2.2 AdminSite

El sitio de administración debió de incorporar la funcionalidad para la carga y administración de alumnos y su relación con las materias. A continuación se describen las vistas y los controladores necesarios para soportar esta funcionalidad.

5.2.3.2.2.1 Views

Las vistas generadas en esta iteración son las necesarias para dar de alta nuevos estudiantes y su relación con las materias.

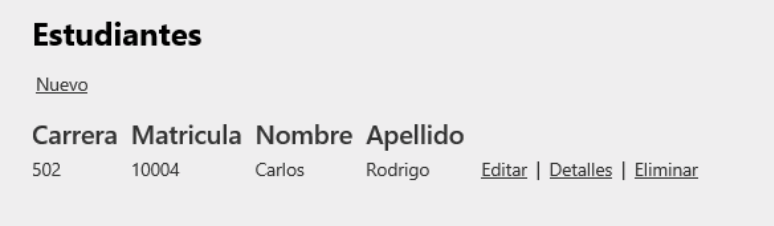

Figura 58 - Lista de estudiantes

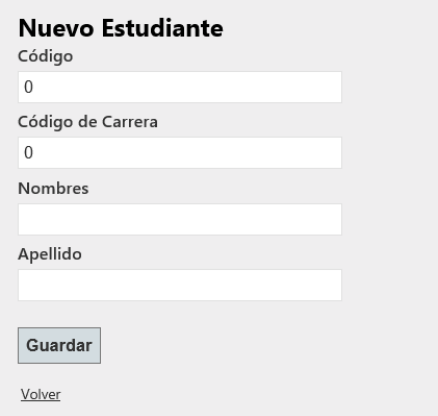

Figura 59 - Vista de Alta y Edición de un estudiante

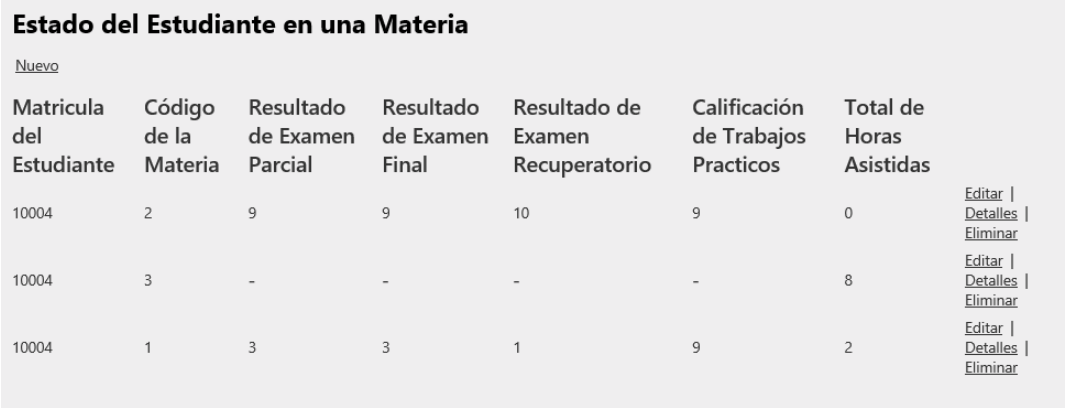

Figura 60 - Lista de relaciones de estudiantes y materias

Figura 61 - Alta y edición de relaciones de estudiantes y materias

También se actualizan las vistas relacionadas con la administración de materias, que muestran la información solicitada para dar por completada la historia *Información de materias*.

Figura 62 - Lista de materias

Figura 63 - Alta y edición de Materia

5.2.3.2.2.2 Controllers

Los controllers que se agregaron son *StudentController* y *StudentCourseRelationship*.

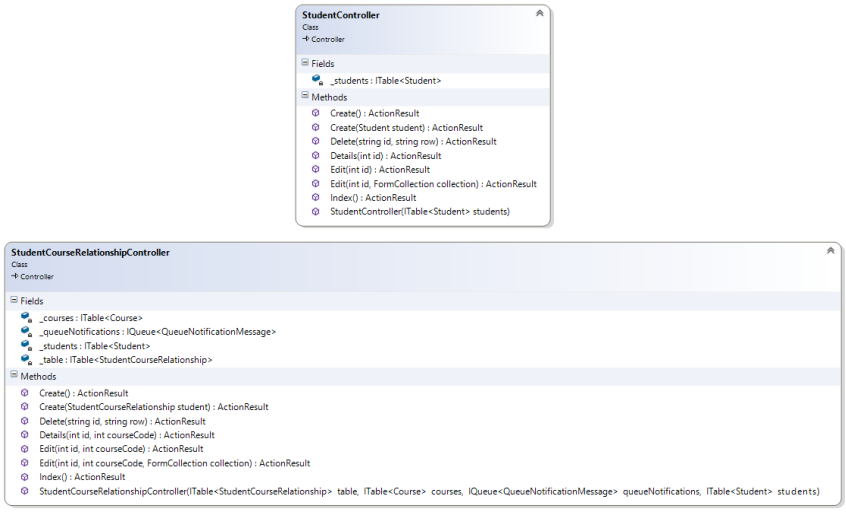

Figura 64 - StudentController y StudentCourseRelationshipController

5.2.4.3. Revisión de la Iteración

Tomando el impulso obtenido en la iteración anterior, al haber obtenido una primera versión de la aplicación para Windows Phone, se pudo conseguir una versión nueva de la misma con más funcionalidades. Estas funcionalidades estaban acotadas al cumplimiento de la historia de usuario *Información de cátedra*, y se han podido alcanzar satisfactoriamente esa meta.

El código fuente resultante de esta Iteración se puede encontrar en el Anexo 3 – Iteración 3.

Para finalizar la Revisión, la tutora del presente TFC completa la grilla de aprobación de historias de usuario.

Historia de Usuario	***Información de cátedra***
Realizado	Realizado, con observaciones de mejora
Observaciones	En la sección *Mi Situación* dentro de una cátedra cambiar: • Porcentaje de asistencias por Porcentaje acumulado de asistencias. En la sección *Descripción* dentro de una cátedra agregar: • Carga horaria semanal • Carga horaria total • Carga horaria dictada En la vista principal • Cambiar *Cátedras* por *Materias* • Separar por *materias en curso*, *materias cursadas*, y *plan de estudio*.
Estado de la Historia	FINALIZADO, con observaciones

5.2.5.Iteración 4

5.2.5.1. Planificación de la Iteración

El resultado obtenido en las iteraciones ha permitido llegar a una cuarta iteración con una versión de la aplicación para Windows Phone y de los servicios en Windows Azure. También se puede afirmar que el ecosistema elegido abarca todo lo esperado.

En esta iteración, y siguiendo el orden de prioridades de las Historias de usuario, le brinda al autor la posibilidad introducirse en otro escenario, las notificaciones. Como se comentó en la sección 4.1.2 Notificaciones Push, Windows Phone provee distintos tipos de notificaciones para que el usuario visualice avisos de la aplicación. Se destaca que la aplicación no necesitar estar ejecutándose para que las notificaciones sean recibidas.

Durante la presente iteración se buscará alcanzar dos historias de usuario, *Aviso para anotarse a examen* y *Aviso de cambio de Asistencia*. Ambas historias tienen una estimación de **5SP**, por lo que se las incluye a ambas en la iteración, además de representar funcionalidades críticas para el usuario.

La estrategia que se usará para la implementación de las notificaciones ya fue nombrada y descripta en la sección Notificaciones Push, e implica el uso de *Queues* de Windows Azure, un *WorkerRole*, y el *Microsoft Push Notification Service*.

Básicamente, se debe crear una *Queue* de Azure que aloje los mensajes a ser enviados. Luego construir un *WorkerRole* que tome los mensajes de la *Queue* y los trasforme al formato que sabe interpretar el *Microsoft Push Notification Service*. Luego el *WorkerRole* se encarga de enviar el mensaje modificado al *Microsoft Push Notification Service*, y este al teléfono en una de las tres formas de notificaciones ya descriptas en la sección Notificaciones Push, Toast, Tile, y Raw Notification.

El sitio de administración recibe información, y si a partir de ella (un cambio en el estado de asistencia del alumno, o la publicación del resultado de un examen, por ejemplo) se debe de enviar una notificación se deposita un mensaje en la Queue de notificaciones.

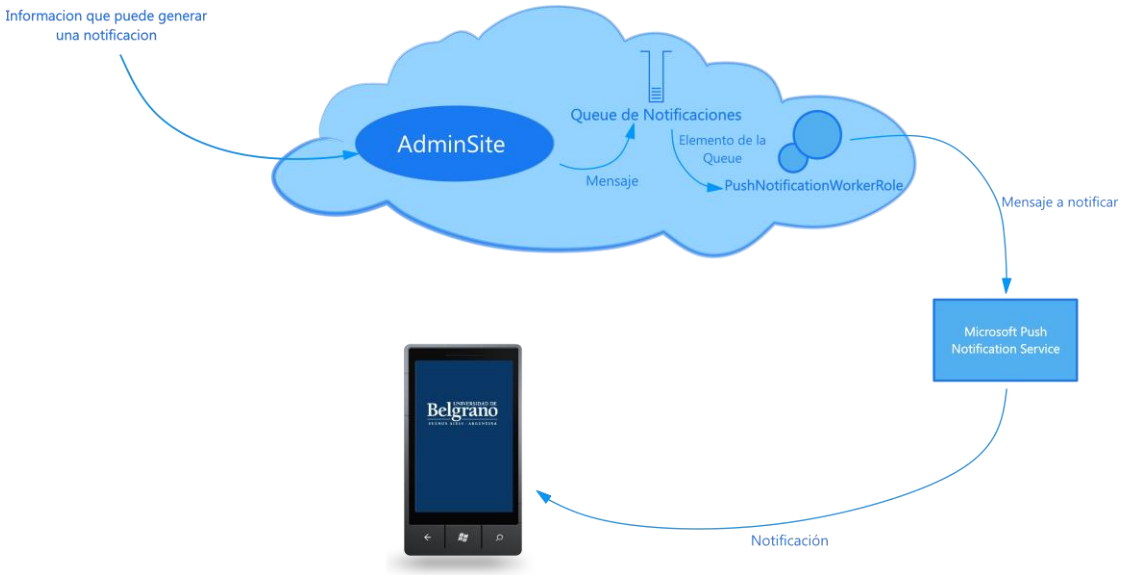

Figura 65 – Estrategia para él envió de notificaciones

Para cumplir con la historia *Aviso para anotarse a examen* resulta de más utilidad un recordatorio que una notificación. Windows Phone provee un mecanismo para agregar recordatorios y alertas, como se describió en la sección 4.1.3.1.1 Scheduled Notifications, y será utilizado para agregar los recordatorios que exige la historia.

5.2.5.1.1. Objetivo de la Iteración

Durante la presente iteración se busca **completar las historias de usuario *Aviso para anotarse a examen* y *Aviso de cambio de asistencia*.**

5.2.5.2. Desarrollo de la Iteración

Continuando con la estrategia presentada en la sección de Planificación de la Iteración, se implementó una *Queue* de Windows Azure y dos tipos de mensajes para la misma, uno para Tile Notifications y otro para Toast Notification. La particularidad de esta implementación de *Queue* es que los mensajes se guardan en formato JSON. De manera que al agregarlos se serializan y al tomarlos de la cola se des serializan para ser utilizados. El motivo de esto es porque las *Queues* en Windows Azure no guardan objetos, sino cadenas.

5.2.5.2.1. SiaService

5.2.5.2.1.1. Model

####### 5.2.5.2.1.1.1. Notifications

Las notificaciones implementadas fueron *TileNotification* y *ToastNotification*, ambas implementan la Interfaz *IPushNotification*, cuya firma consisten en las propiedades Type y Payload. La primera permite saber el tipo de Notificación, y la segunda es el formato en que la notificación debe enviarse al MPNS. Este formato es un XML que difiere para cada notificación.

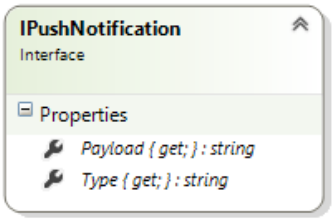

Figura 66 - IPushNotification

La implementación del *Tile Notification* solo expone las propiedades de la parte trasera del tile, BackBackgroundImage, BackContent, BackTile. Estas fueron desarrolladas en la sección Notificaciones Tile.

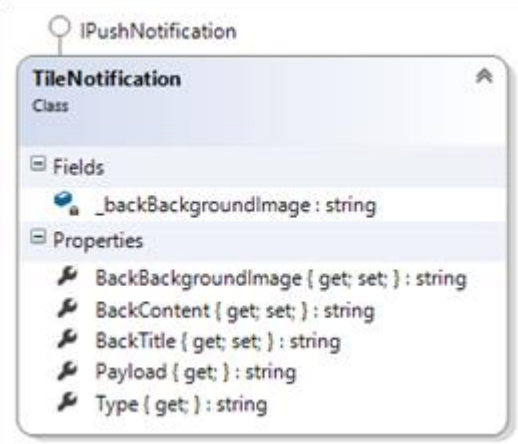

Figura 67 - TileNotification

El formato para enviar una notificación Tile es el siguiente:

```
"<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
"<wp:Notification xmlns:wp=\"WPNotification\">" +
   "<wp:Tile >" +
      "<wp:BackgroundImage>" +
      "</wp:BackgroundImage>" +
      "<wp:Count>" +
      "</wp:Count>" +
      "<wp:Title>" +
      "</wp:Title>" +
      "<wp:BackBackgroundImage>{0}</wp:BackBackgroundImage>"
      "<wp:BackTitle>{1}</wp:BackTitle>" +
      "<wp:BackContent>{2}</wp:BackContent>" +
   "</wp:Tile> " +
"</wp:Notification>"
```

La implementación de *Toast Notification* expone todas las propiedades vistas en la sección 4.1.2.2 Notificaciones Toast, con la diferencia que se la propiedad *Params* se nombró Page, cuyo objetivo es seleccionar la página de la aplicación Mobile a mostrar.

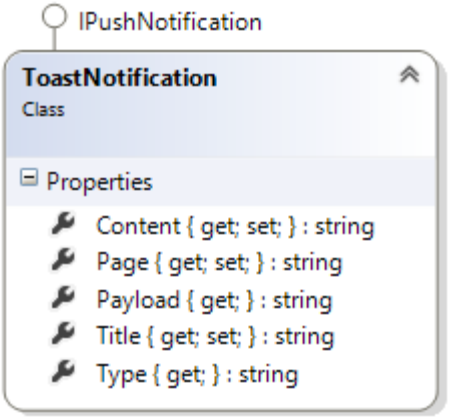

Figura 68 - ToastNotification

El formato de una notificación Toast es:

```
"<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                    "<wp:Notification xmlns:wp=\"WPNotification\">" +
                      "<wp:Toast>" +
                        "<wp:Text1>{0}</wp:Text1>" +
                        "<wp:Text2>{1}</wp:Text2>" +
                        "<wp:Param>{2}</wp:Param>" +
                      "</wp:Toast>" +
                    "</wp:Notification>"
```

5.2.5.2.1.1.2. Notification Message

Como muestra la Figura 65, en la *Queue* de notificaciones se deposita mensajes que representan las notificaciones que se desean enviar a los dispositivos Mobile.

Para que la estrategia sea exitosa, el mensaje a depositar en la *Queue* de notificaciones debe poseer toda la información necesaria para ser enviada la notificación a la brevedad. Para cumplir estas premisas se implementa la entidad *QueueNotificationMessage*.

La entidad *QueueNotificationMessage* cuenta con la propiedad Payload que es el cuerpo de la notificación, Type que responde al tipo, Priority que representa la prioridad con la que la notificación debe ser enviada, y Destinatary que es la identificación única que le provee el Microsoft Push Notification Service a la aplicación Windows Phone. Esta última será explicará en profundidad en la sección 5.2.5.2.4.1 Views.

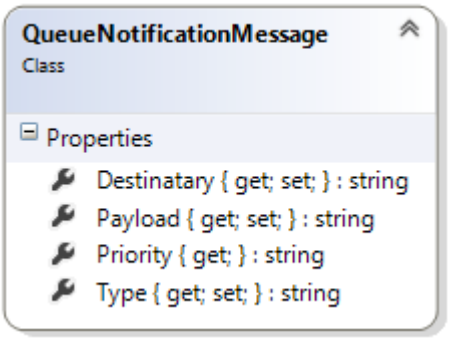

Figura 69 - QueueNotificationMessage

5.2.5.2.1.1.3. Notification Queue

Anteriormente se ha definido la Interfaz que representa una *Table* en Azure como ITable<T>, en este caso se realizará el mismo procedimiento para una *Queue* de Azure.

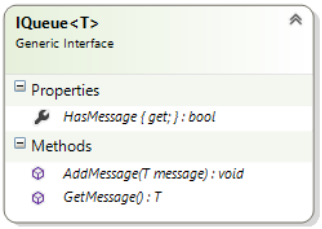

Figura 70 - IQueue

NotificationQueue es la entidad que implementa la Interfaz IQueue.

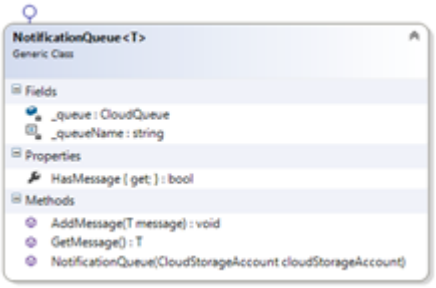

Figura 71 - NotificationQueue

La entidad *NotificationQueue* mediante su método AddMessage almacena las notificaciones a enviar, y utiliza GetMessage para entregar el primer mensaje que posee.

Para simplificar y optimizar el uso de las Queue de Azure, al momento de almacenar los mensajes, representados por la entidad QueueNotificationMessage, esta entidad es serializada a formato JSON, y luego des serializada al momento que se solicita un mensaje en la Queue. Esta labor se realiza gracias a la clase JsonSerializationHelper.

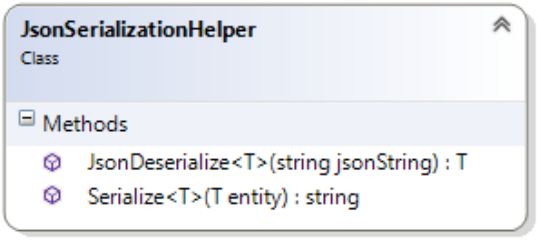

Figura 72 - JsonSerializationHelper

5.2.5.2.1.1.4. Student

La entidad Student incorpora a sus propiedades el DeviceUri, la URI que el Microsoft Push Notification Service entrega a la aplicación cuando se registre para recibir notificaciones, esta información corresponde que se almacene junto con el resto de información del estudiante.

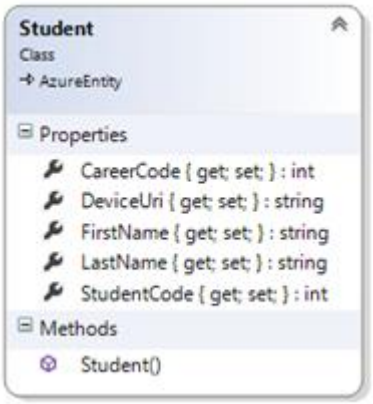

Figura 73 - Student

5.2.5.2.2. PushNotificationSenderRole

Luego de haber definido la implementación de las notificaciones y la forma de almacenarlas, es necesario enviarlas. Para ello, como la estrategia elegida lo propone, se implementa un WorkerRole con el nombre PushNotificationSenderRole. Como se describió en la sección 4.2.1.3.2.1 Windows Azure Roles, un Worker Role está orientado al procesamiento de datos y es básicamente una aplicación que está ejecutando por tiempo indefinido el método Run. Dentro de ese método es donde se realiza la obtención del mensaje y su posterior envío.

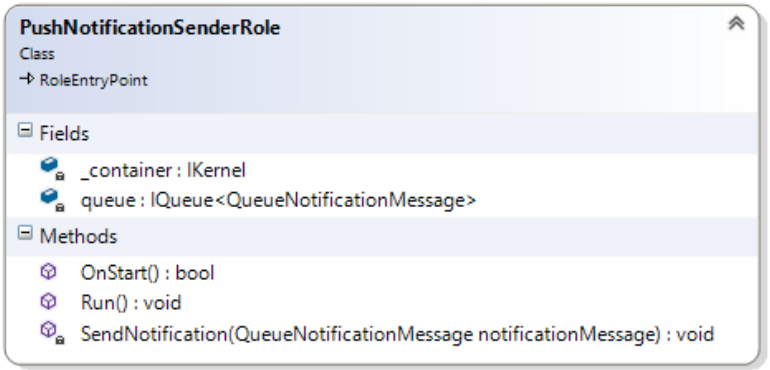

Figura 74 - WorkerRole PushNotificationSenderRole

Cada vez que se obtiene un mensaje de la *NotificationQueue* y es entregado como parámetro al método *SendNotification* para ser enviado, que en caso de no ser capaz de hacerlo,

vuelve a depositar el mensaje en la cola. Para conocer los detalles de implementación de este mecanismo, puede consultarse el Anexo 4 – Iteración 4.

5.2.5.2.3. AdminSite

El sitio de administración será el encargado de agregar los mensajes a la *Queue* de notificaciones. Esto se debe a que es quien contiene la información sobre el alumno y su relación con las materias.

Se han agregado en la sección Estudiante-Materia dos acciones que será disparadores de notificaciones, *Asignar Calificación* y *Asignar Horas Cursadas*.

Figura 75 - Lista de la relación Estudiante-Materia con las acciones que se pueden realizar

La acción *Asignar Horas Cursadas* dirige a un formulario en el cual se agregan las horas asistidas del alumno a la materia, las mismas son incrementales.

Figura 76 - Formulario para asignar horas asistidas

Cuando una se produce un cambio en la situación de asistencia que responde a los criterios de aceptación de las historia *Aviso de cambio de asistencia*, se envía una *Tile Notification*. Esta notificación muestra el nombre de la materia, el mensaje “Nuevo estado de asistencia”, y el fondo del Tile cambia de color según el estado: Rojo cuando la situación es baja, Amarillo cuando es comprometida, y Verde cuando es buena.

La visualización de estas notificaciones se muestra en la sección 5.2.5.2.4.4 Visualización de notificaciones.

En cuanto a la acción *Asignar Calificación*, la misma permite seleccionar el tipo de examen y asignar una calificación. Los tipos de examen son: Parcial, Recuperatorio, Final, y Presentación de Trabajos Prácticos.

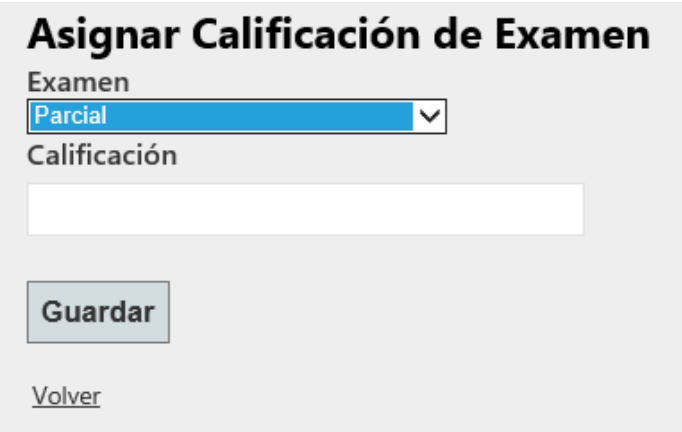

Figura 77 - Formulario para asignar calificación

Al asignar la calificación, también se aprovecha para enviar dos notificaciones a dispositivo Mobile. Una notificación Tile con el tipo y resultado del examen, y la materia a la cual pertenece. Y una notificación Toast con la misma información.

Ambas notificaciones se muestran en la sección 5.2.5.2.4.4 Visualización de notificaciones.

5.2.5.2.4. Aplicación Windows Phone

Como se explica en la sección 4.1.2 Notificaciones Push, la aplicación Mobile debe contactarse con el Microsoft Push Notification Service para suscribirse y poder empezar a recibir notificaciones. Además, para cumplir con la historia *Aviso para anotarse a examen* se deben agregar recordatorios para los exámenes cuando estos tengan una fecha confirmada.

5.2.5.2.4.1. Views

Con la necesidad de suscribirse al MPNS se creó la View *SettingsView*, que brinda al usuario de la aplicación la posibilidad de recibir notificaciones o no. Ver Figura 79.

Para acceder a la vista *SettingsView* se agregó en *MainPage* una ApplicationBar con el ítem “Configuración”, que al ser presionado dirige a la vista SettingsView.

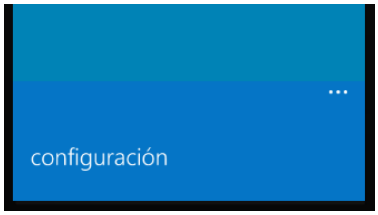

Figura 78 - Ítem Configuración en vista MainPage

La acción de guardar que puede observarse en la Figura 79, dispara una llamada al MPNS. Una vez que este responde, entrega a la aplicación un URI, con el cual el MPNS va a identificar el dispositivo a cual enviar la notificación y la aplicación responsable de ella.

Figura 79 - SettingsView

5.2.5.2.4.2. ViewModels

En esta iteración solo *SettingsViewModel* se ha agregado a los ViewModels existentes.

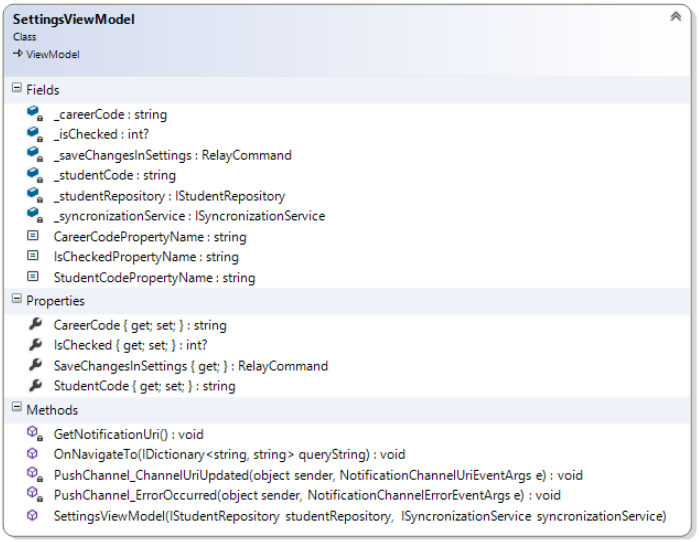

Figura 80 - SettingsViewModel

SettingsViewModel a través del método GetNotificationUri se encarga de obtener el URI del MPNS, y con la ayuda del *SyncronizationService* envía la URI obtenida al SiaService para que la almacene en la información del estudiante.

5.2.5.2.4.3. Services

En esta iteración surgió la necesidad de registrar recordatorios en el dispositivo Mobile para las fechas de examen, esta tarea se lleva a cabo cada vez que se accede a una materia. Si al descargar la información se encuentra una fecha de examen, entonces se crean los recordatorios para ellas según lo establece el criterio de aceptación de la historia *Aviso para anotarse a un examen*. Ya no solo se actualiza la información de una cátedra, sino que también se registran los recordatorios de la misma. Por lo que resulta necesario contar con una entidad que gestione la información de una cátedra, *StudentInformationManager*.

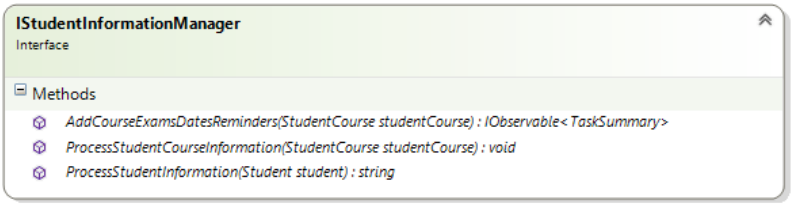

Figura 81 - IStudentInformationManager

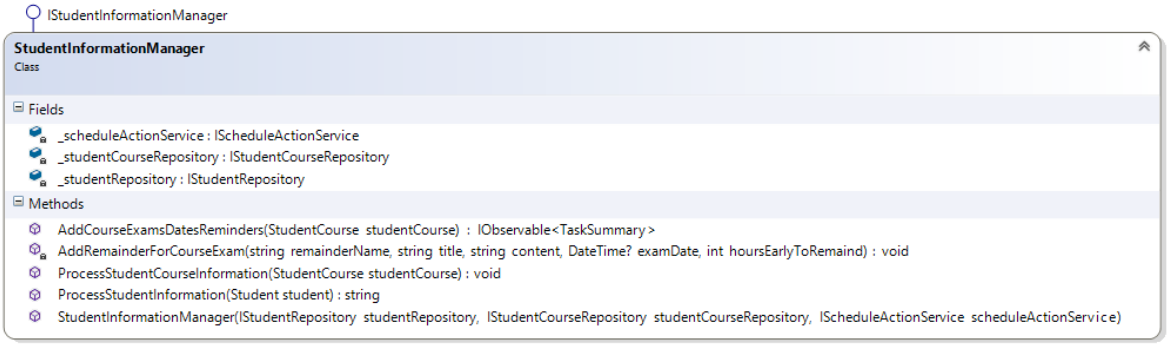

Figura 82 - StudentInformationManager

Esta entidad no solo agrega los recordatorios, sino que también es la que actualiza la información del alumno y sus materias en el IsolatedStorage.

El servicio *SyncronizationService* cuenta ahora con un método más, encargado de asignar la URI obtenida del MPNS al estudiante.

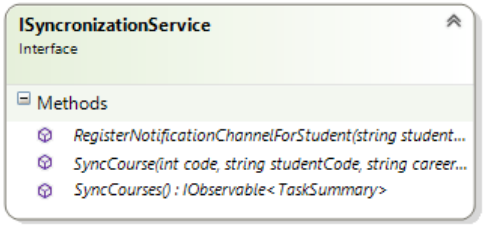

Figura 83 - ISyncronizationService

5.2.5.2.4.4. Visualización de notificaciones

Como se adelantó en las secciones anteriores, en esta sección se mostraran de qué manera se visualizan las notificaciones y recordatorios.

Las notificaciones que se envían cuando se realiza un cambio en el estado de asistencia se muestran de la siguiente manera.

Figura 84 - Visualización de la notificación de cambio de asistencia

En la figura anterior se pueden apreciar los estados mencionados en 5.2.5.2.3 AdminSite, para la acción *Asignar horas cursadas*.

En la se ven ambas notificaciones al mismo tiempo. La notificación Tile informa la materia, el examen, y su resultado. De igual forma lo hace la notificación Toast, pero ésta tiene la capacidad que la tocarla inicia la aplicación y nos dirige directamente a la pagina de la cátedra de la que se informó el resultado del examen.

Figura 85 - Visualización de notificaciones enviadas al Asignar Calificación

Por ultimo se muestran a continuación uno de los recordatorios que se crean cuando la cátedra posee una fecha de examen.

Figura 86 - Recordatorio de compromiso académico

5.2.5.3. Revisión de la Iteración

En esta iteración se consiguió un avance muy importante, la implementación de la arquitectura necesaria para el envío de notificaciones de tipo *Tile* y *Toast*. Además se lograron agregar recordatorios para los exámenes. Ambas funcionalidades aportan al estudiante un valor agregado y le son de gran utilidad.

El avance conseguido permitió incluir dentro de la iteración la historia *Aviso de publicación de calificación de examen*. La cual como se pudo observar en el desarrollo, fue terminada junto con las planificadas.

Surgieron dificultades al momento del envío de las notificaciones y la creación de las *Queue*, las lecciones aprendidas son:

- Si se envía una notificación mal formada, el canal de comunicación entre el MPNS y el dispositivo móvil queda bloqueado. Es necesario reinstalar la aplicación y volver a suscribirse al MPNS. Sino las notificaciones nunca llegan a los dispositivos.
- Los nombres con los que se crean las *Queue* deben estar compuestos solo por letras en minúscula. De otra manera se obtendrá un error.
- Es necesario que si una notificación cambia el *BackBackgroundImage* del Tile, la próxima notificación también lo haga. Caso contrario se mostrará el mensaje con una imagen de fondo que puede no corresponder con el mensaje que se quiere dar.

Al terminar esta iteración se han completado todas las historias de prioridad alta, y una de prioridad baja.

El código fuente de la versión de la aplicación obtenida en esta iteración se encuentra en el Anexo 4 – Iteración 4

A continuación se presenta la tabla de fin de iteración que el líder de proyecto completa con sus observaciones.

Historia de Usuario *Aviso para anotarse a examen*	
Realizado	Objetivos alcanzados parcialmente
Observaciones	Cambios a realizar: Reemplazar “Presentación de Trabajos Prácticos de “ por “Presentación TPs *{Materia}*”
Estado de la Historia	FINALIZADO, con observaciones

Historia de Usuario *Aviso de cambio de asistencia*	
Realizado	Objetivos alcanzados parcialmente
Observaciones	Cambios a realizar: Cambiar el fondo amarillo con letra blanca por un color que permita una fácil visualización
Estado de la Historia	FINALIZADO, con observaciones

Historia de Usuario *Aviso de publicación de calificación de examen*	
Realizado	Objetivos alcanzados

Observaciones	No se observan cambios a realizar
Estado de la Historia	FINALIZADA, no se detectan errores

5.2.6.Iteración 5

En la iteración 3 y la iteración anterior, el líder de proyecto realizó una serie de sugerencias y observaciones. Estas serán incluidas dentro de la planificación de la iteración junto con las historias de usuario que se decidan desarrollar.

5.2.6.1. Planificación de la Iteración

Siguiendo el orden de prioridades de las historias de usuario, y dado que las de alta prioridad ya fueron desarrolladas, las siguientes historias de prioridad media son *Inscripción a examen final* y *Acceso a la Aplicación*.

Como se adelantó, también se deben de incluir en esta iteración las sugerencias y observaciones que el líder de proyecto describió al final de la iteración 3. Estas sugerencias y observaciones son:

En la sección *Mi Situación* dentro de una cátedra cambiar:

- Porcentaje de asistencias por Porcentaje acumulado de asistencias.

En la sección *Descripción* dentro de una cátedra agregar:

- Carga horaria semanal
- Carga horaria total
- Carga horaria dictada

En la vista principal

- Cambiar *Cátedras* por *Materias*
- Separar por *materias en curso*, *materias cursadas*, y *plan de estudio*.

En la iteración anterior, el líder de proyectos sugirió también:

- Cambiar el fondo amarillo con letra blanca por un color que permita una fácil visualización.
- Reemplazar “Presentación de Trabajos Prácticos de “ por “Presentación TPs *{Materia}*”

Estas correcciones se estiman en **2SP**, con lo que se completaría el tiempo que dura la iteración. Y con la suma de las historias de usuario hacen un total de **10SP**.

5.2.6.1.1. Objetivos de la iteración

En resumen, la presente iteración busca finalizar la historia ***Inscripción a examen final*** y realizar las correcciones que indicó el líder de proyectos en la iteración 3.

5.2.6.2. Desarrollo de la Iteración

Para cumplir la historia *Inscripción a examen final* se agregó un método en el SiaService que recibe los datos del alumno y la materia. Se valida si el alumno está en condiciones y devuelve un mensaje de aprobación o rechazo a de la inscripción.

En la aplicación para Windows Phone, dentro de la ApplicationBar en de la *CourseView* se agregó un ítem de menú para realizar la acción de inscripción a examen final.

En cuanto a la historia *Acceso a la aplicación*, se resuelve de manera sencilla. Al ingresar a la aplicación se busca en el *IsolatedStorage* información sobre un alumno, en caso de no encontrarla se visualiza una vista que pide como información el número de matrícula, número de carrera, y la clave de seguridad del alumno. Con esa información, se hace un llamado al SiaService para descargar la información relacionada con el alumno.

Las correcciones propuestas por el líder de proyectos al final de la iteración 3, en las próximas secciones se mostraran los cambios implementados.

5.2.6.2.1. Aplicación Windows Phone

5.2.6.2.1.1. Views

Se agregó la vista para el ingreso de las credenciales del alumno, *UserCredentialsView.*

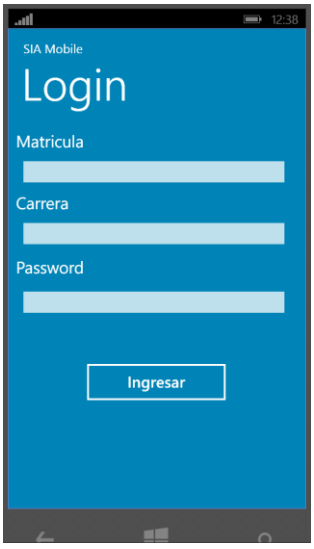

Figura 87 - UserCredentialsView

Como se adelantó, la acción de inscripción se implementa como un ítem de menú en la ApplicationBar de la vista *CourseView*, y luego se muestra un mensaje donde se informa la aprobación o rechazo de la inscripción.

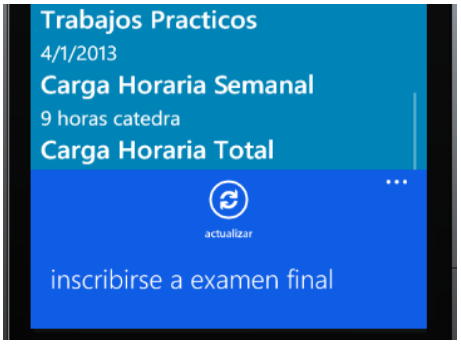

Figura 88 - Acción para inscribirse a examen final

Como respuesta se puede obtener los siguientes mensajes.

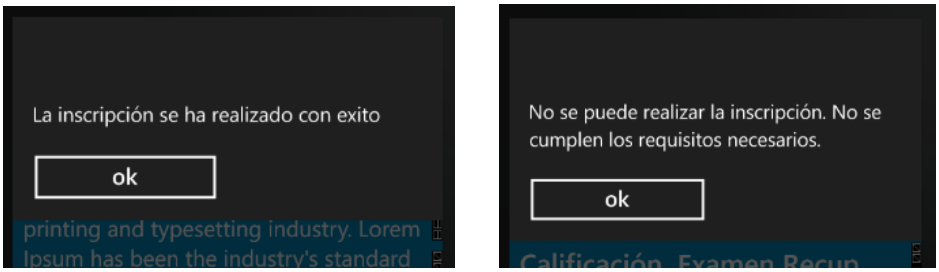

Figura 89 - Mensajes de respuesta de la acción Inscribirse a examen final

A continuación se muestran las correcciones realizadas en las secciones *Mi Situación* y *Descripción* de la vista CourseView.

Figura 90 - Correcciones realizadas en las secciones Mi Situación y Descripción de la vista CourseView

Para cumplir con la última observación realizada por el líder de proyectos, se cambió el título “Cátedras” por “Materias” en la MainView, donde se muestran solo las materias que el alumno está cursando actualmente.

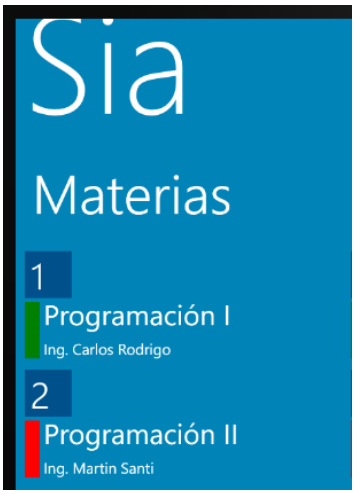

Figura 91 - Cambio en el título del listado de materias

Como se puede observar, se agregó la funcionalidad de agrupar por año las materias. Y al presionar cada año se ofrece un listado de los años de la carrera para navegar hacia las materias correspondientes a ese año.

Figura 92 - Listado de años de carrera para navegar las materias

Además se agregó en la *ApplicationBar* la funcionalidad “ver todas”, que redirige a la vista *AllCoursesView*. Esta vista posee tres listados de materias, *En Curso*, *Cursadas*, y *Plan de estudios*.

Figura 93 - Acción "Ver todas"

A continuación se muestra como se visualizan las materias en los tres listados nombrados anteriormente.

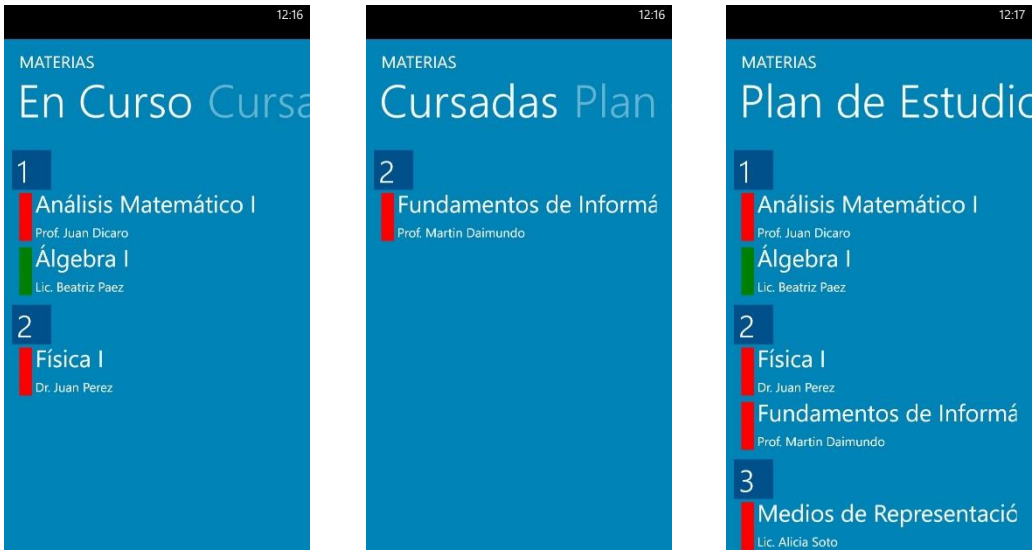

Figura 94 - Lista de materias por En Curso, Cursadas, y Plan de Estudio

5.2.6.2.1.2. ViewModels

En esta iteración se agregó el ViewModel *AllCoursesViewModel* que contiene tres colecciones de materias, cada una corresponde a una de las listas de materias que se muestran en la vista *AllCoursesView*.

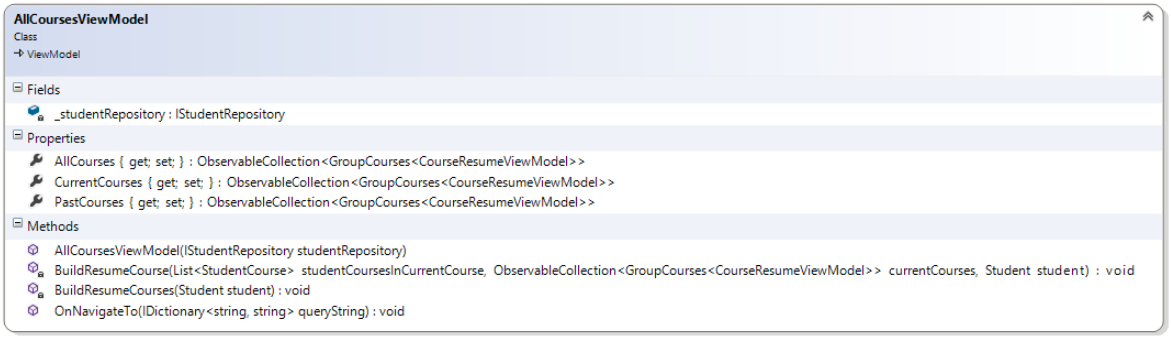

Figura 95 - AllCoursesViewModel

Se toma del *IsolatedStorage* de la aplicación la lista completa de materias. Para la lista de materias en curso, se toman aquellas que están marcadas como “en curso”. Para las materias

cursadas, se toman aquellas que posean resultado de examen parcial o recuperatorio, cuenten con un porcentaje de asistencias distinto de cero, y no estén marcadas como “en curso”. En el último caso, las materias correspondientes al plan de estudio son todas las materias.

También se agregó en el ViewModel *CourseViewModel* el command *CheckForFinalExam,* que realiza la acción de inscribirse a examen final.

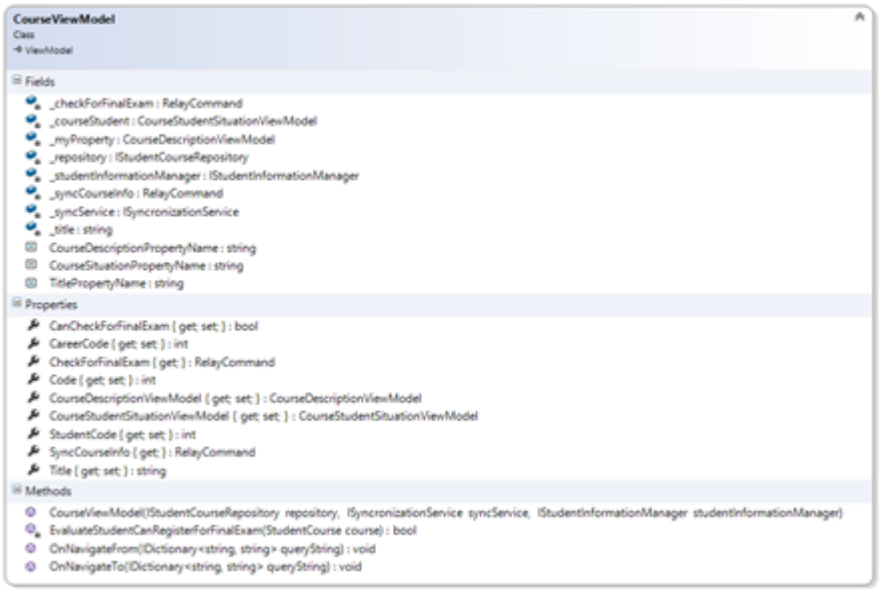

Figura 96 - Clase CourseViewModel actualizada.

5.2.6.2.1.3. Services

En cuanto a los servicios en la aplicación móvil, se adaptaron a la funcionalidad de *inscribirse a examen final*. En el servicio *SyncronizationService* se agregó el método *RegisterNotificationChannelForStudent*.

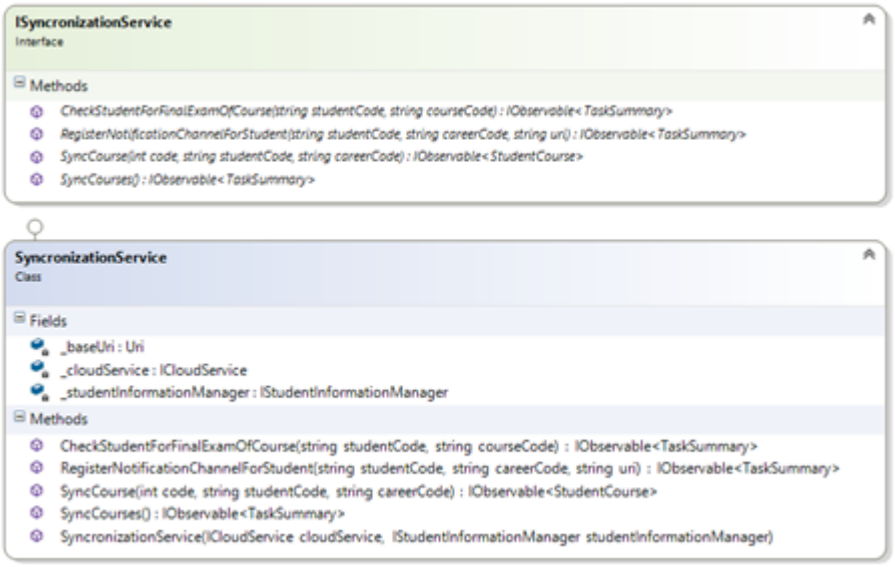

Figura 97 - Interfaz ISyncronizationService y su implementación SyncronizationService

En el servicio *SyncronizationService* interactúa con el servicio *CloudService* para realizar la acción de *inscribirse a examen final*, por lo que el último también implementa un nuevo método para comunicarse con el SiaService y llevar acabo la acción.

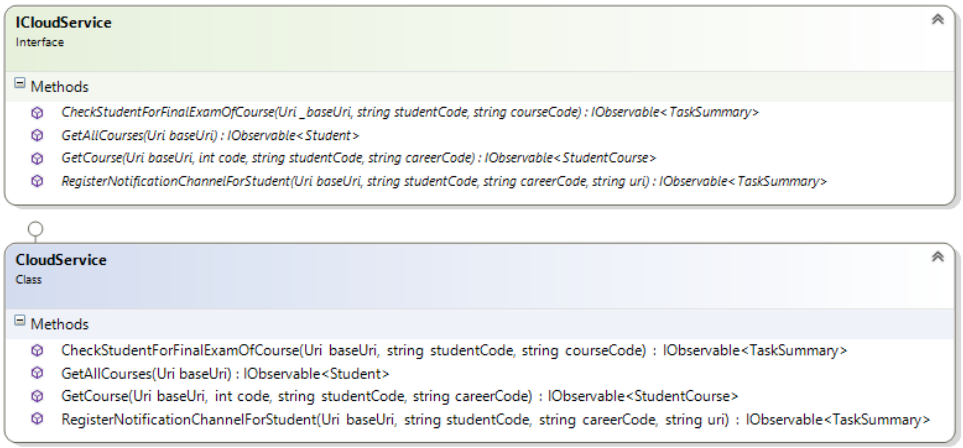

Figura 98 - Interfaz ICloudService y su implementación CloudService

5.2.6.2.1.4. Visualización de notificaciones

En esta sección se realizan las modificaciones necesarias para cumplir con las observaciones que el líder de proyectos detallo al final de la iteración anterior.

Se cambió el color de fondo para la notificación de estado de asistencias comprometido, de amarillo a un tono anaranjado.

Figura 99 - Nuevo color para la asistencia en estado intermedio

También se cambió el mensaje para el recordatorio de la presentación de trabajos prácticos.

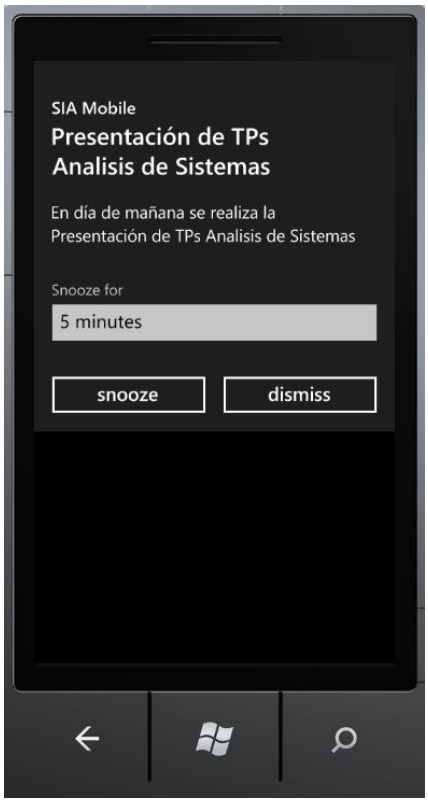

Figura 100 - Corrección en el recordatorio para la presentación de TPs

5.2.6.2.2. SiaService

El servicio SiaService también ha incorporado un método para realizar la acción necesaria para cumplir con la historia *Inscripción a examen final*. El mismo se llama *RegisterStudentForFinalExam*.

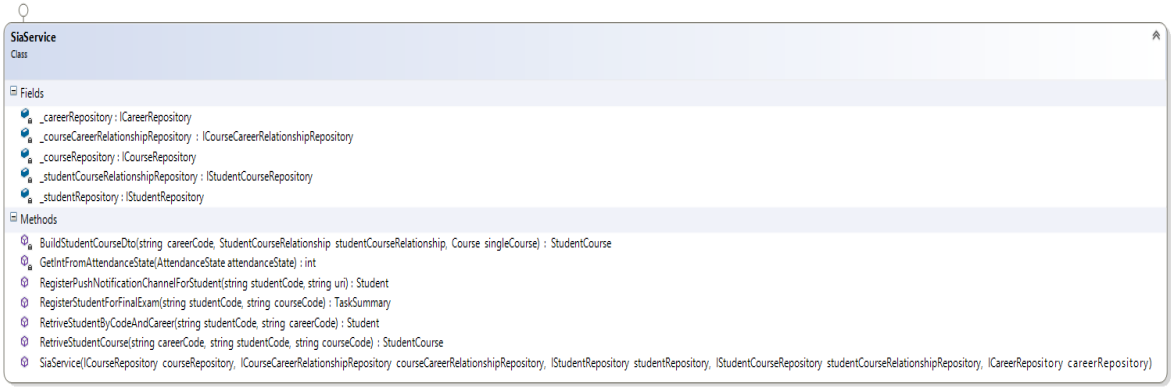

Figura 101 - Actualización en la interfaz ISiaService y su implementación SiaService

Este método valida si el alumno está en condiciones de inscribirse en el examen final de una materia en particular o no. Y envía como respuesta los mensajes que se describieron en la sección 5.2.6.2.1.1 Views de esta iteración.

5.2.6.2.2.1. Model

Durante la presente iteración, el modelo de datos ha sufrido cambios menores. En la entidad *StudentCourseRelationship* se han agregado dos propiedades, *FinalExamInscripted* e *IsCurrentCourse*. La primera propiedad cumple la funcionalidad de indicar si el alumno ya está inscripto al examen final de la materia, y la segunda indica si está cursando la materia actualmente.

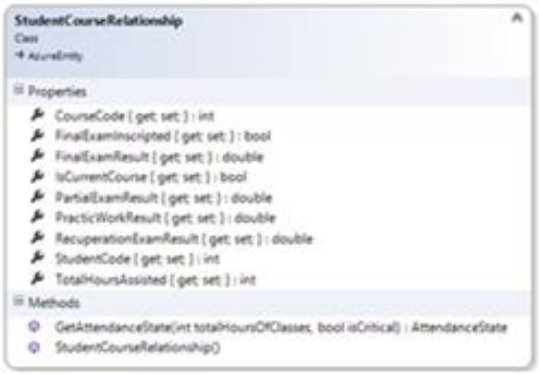

Figura 102 - StudentCourseRelationship

La entidad *Course* también se ha modificado para cumplir con los objetivos de esta iteración, en este caso las correcciones que el líder de proyectos propuso. Se agregaron las propiedades *TotalHoursInAWeek* y *TotalHoursDictated*, que representan el total de horas semanales y las horas dictadas a la fecha.

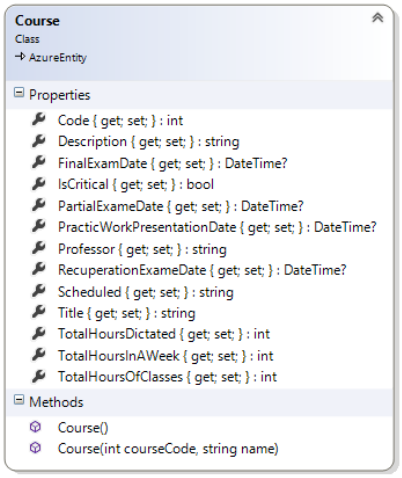

Figura 103 - Propiedades TotalHoursInAWeek y TotalHoursDictated en Course

5.2.6.2.3. AdminSite

En el sitio de administración se han actualizado los formularios de carga Materias y Asignación de Materias a estudiantes para que pueda cargar la cantidad de horas semanales y dictadas de una materia, como también la posibilidad de asentar manualmente que un alumno está inscripto en una materia.

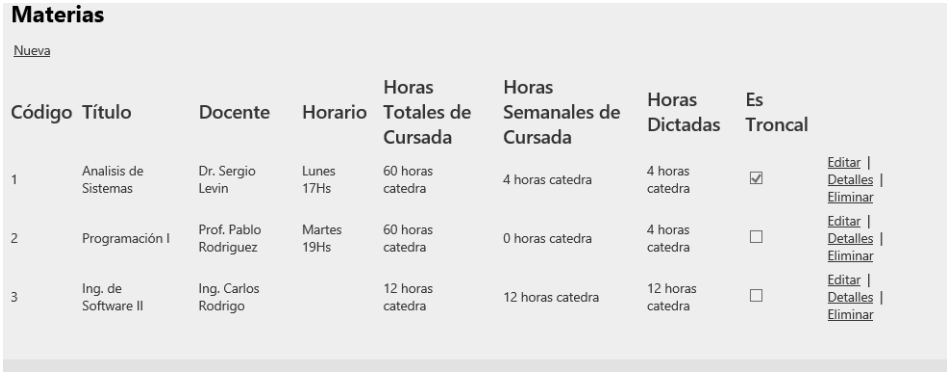

Figura 104 - Lista de materias con horas semanales de cursada visible

Estado del Estudiante en una Materia

Nuevo

Matricula del Estudiante	Código de la Materia	Resultado de Examen Parcial	Resultado de Examen Final	Resultado de Examen Recuperatorio	Calificación de Trabajos Practicos	Total de Horas Asistidas	Inscripto a Examen Final	Cursando esta materia	
10004	2	9	9	10	9	0	No	Si	Asignar Calificación Asignar Horas Cursadas Editar | Detalles | Eliminar
10004	3	-	-	-	-	8	No	Si	Asignar Calificación Asignar Horas Cursadas Editar | Detalles | Eliminar
10004	1	3	3	1	9	2	No	No	Asignar Calificación Asignar Horas Cursadas Editar | Detalles | Eliminar

Figura 105 - Lista de materias asignadas a un alumno y descripción, si está inscripto y cursando la materia

5.2.6.3. Revisión de la Iteración

En esta iteración no se produjeron grandes cambios, se cumplió con la historia *Inscribirse a examen final* y se llevaron a cabo los comentarios que el líder de proyectos realizó al final de la iteración anterior.

Además se pudo desarrollar la última historia de usuario en la lista, *Acceso a la aplicación*. De esta manera se da por finalizado el alcance y se procede a la última etapa que establece la metodología utilizada, Entrega del producto.

Antes, y como se realizó en todas las iteraciones anteriores, se pone a disposición del líder de proyectos la ficha de revisión de la iteración para que describa sus impresiones sobre el resultado obtenido al final de esta iteración.

Historia de Usuario	*Inscripción a examen final*
Realizado	Objetivos alcanzados
Observaciones	En esta iteración no se encuentran ni correcciones ni sugerencias de mejora.
Estado de la Historia	FINALIZADA, no se detectan errores

Historia de Usuario	*Acceso a la aplicación*

Realizado	Objetivos alcanzados
Observaciones	En esta iteración no se encuentran ni correcciones ni sugerencias de mejora.
Estado de la Historia	FINALIZADA, no se detectan errores

5.3. Entrega del producto

Esta etapa está dedicada a la entrega y despliegue de la solución de software obtenida durante la sección 5.2 Construcción de software.

Como se explicará en la próxima sección, la entrega de la solución consiste en el código fuente obtenido a lo largo de las cinco iteraciones, pero además se debe realizar el empaquetado de la aplicación para Windows Phone y de los servicios de Windows Azure. Para esto se describen los pasos necesarios para obtener los paquetes y su posterior despliegue en ambas plataformas.

En el caso de Windows Azure, se debe de generar el paquete para que sea desplegado en los servidores de la plataforma.

Y en cuanto a la aplicación de Windows Phone, debe de formar parte del *Marketplace* de aplicaciones del sistema operativo, para que los alumnos puedan descargarla en sus teléfonos inteligentes.

5.3.1. Entrega final

Como se adelantó, la entrega final se compone de tres partes: Entrega de código fuente, Publicación de los Roles de Windows Azure, y Publicación de la aplicación Windows Phone en el *Marketplace* de aplicaciones.

5.3.1.1. Entrega de código fuente

Al final de cada iteración, se hace referencia a un documento anexo que contiene el código fuente generado en la iteración. Las iteraciones son iterativas e incrementales, por lo que la última iteración da como resultado la versión completa de la solución de software generada.

La entrega final del código fuente del software obtenido en el presente Trabajo Final de Carrera está documentado en el Anexo 5 – Iteración 5.

5.3.1.2. Publicación de los Roles de Windows Azure

La publicación consisten en generar una paquete que contenga el código compilado de los roles y también la descripción y configuración del servicio que se desea desplegar en Windows Azure, como se describió en la sección 4.2.1.3.2.4 Empaquetado del servicio.

Con la ayuda de Visual Studio 2012, se puede generar y publicar el paquete en la plataforma de manera rápida, pero se debe contar con una cuenta de Windows Azure para poder

realizar esta tarea. Para obtener la cuenta se debe contar con una cuenta de Windows Live y una tarjeta de crédito para realizaran el pago por los servicios utilizados. Windows Azure ofrece una suscripción de prueba por tres meses.

Para generar la el paquete y publicarlo a la plataforma Windows Azure desde Visual Studio se debe utilizar la herramienta de *Publish.* Para ello se debe de posicionar sobre el proyecto de Windows Azure, hacer clic derecho, y seleccionar la herramienta mencionada.

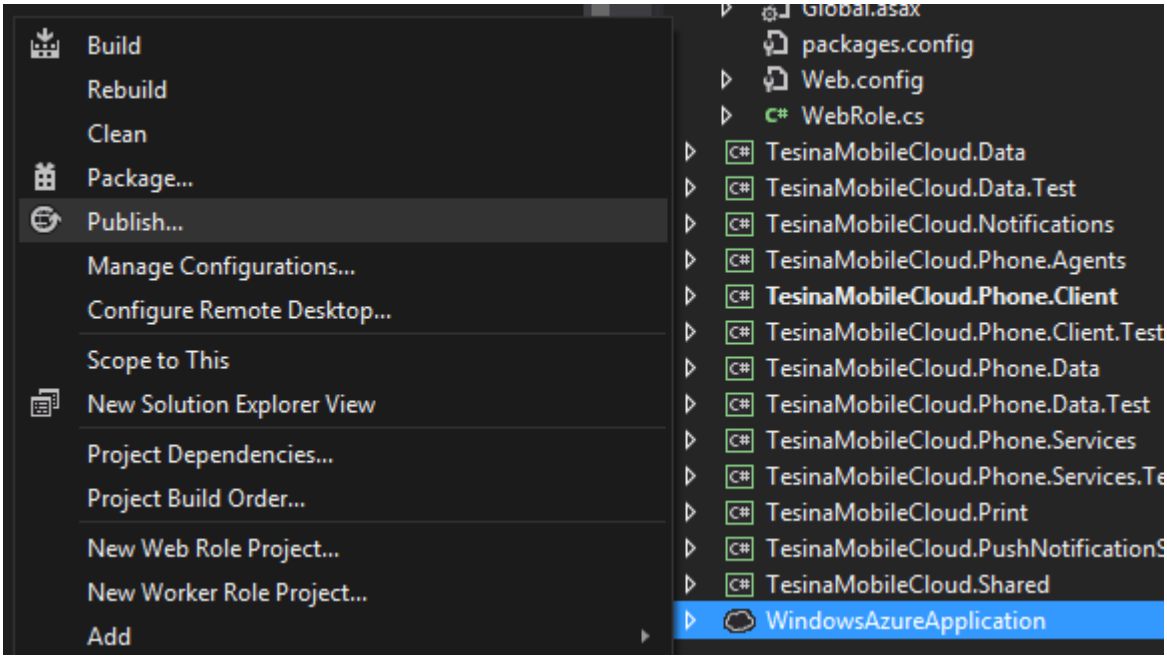

Figura 106 - Selección de herramienta Publish para publicar los roles de Windows Azure desde Visual Studio

Una vez seleccionada, se despliega la aplicación que realiza el empaquetado y publicación de los Roles de Windows Azure. El primer paso es cargar la información de la cuenta de Windows Azure que se posee, utilizando la opción *Sign in to download credentials* que se muestra en la Figura 107. Esto abrirá una página web donde se deben ingresar la información de la cuenta de Windows Azure. Una vez ingresada esta información se descarga un archivo con extensión *.publishsettings,* que contiene la información de la suscripción de Windows Azure generada con esa cuenta.

El siguiente paso es utilizar este archivo para cargar la información en la aplicación de publicación. Para esto se debe utilizar el botón *Import* que se ve en la Figura 107 y seleccionar el archivo que se descargó anteriormente.

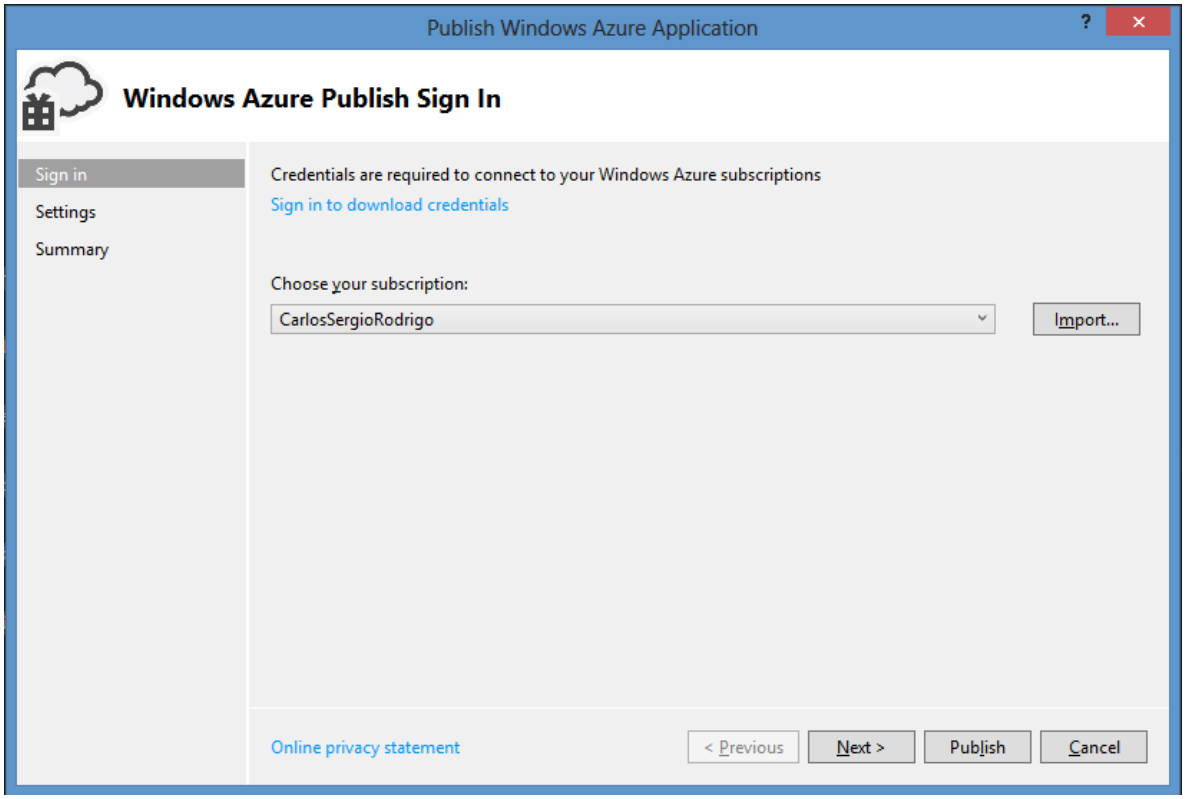

Figura 107 - Herramienta de publicación de aplicaciones de Windows Azure

Luego de haber cargado el archivo *.publishsettings* continuar con el siguiente paso mediante el botón *Next*. Esta acción abre un cuadro de dialogo que solicita el nombre con el que se va nombrar el servicio que contendrá a los Roles generados, y además solicita el área geográfica donde se desean desplegar, como se observa en la Figura 108. Luego de completar esta información se vuelve a presionar el botón *Next* para dar lugar al último paso necesario para la publicación.

Para finalizar, solo es necesario hacer clic en el botón *Publish* que se muestra en la Figura 109

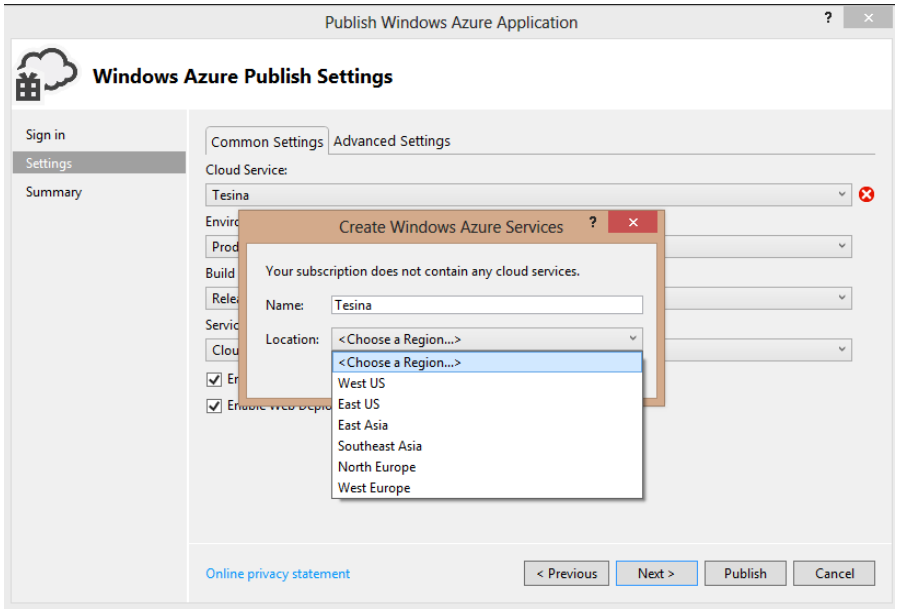

Figura 108 - Selección de nombre y ubicación geográfica para el despliegue de los Roles

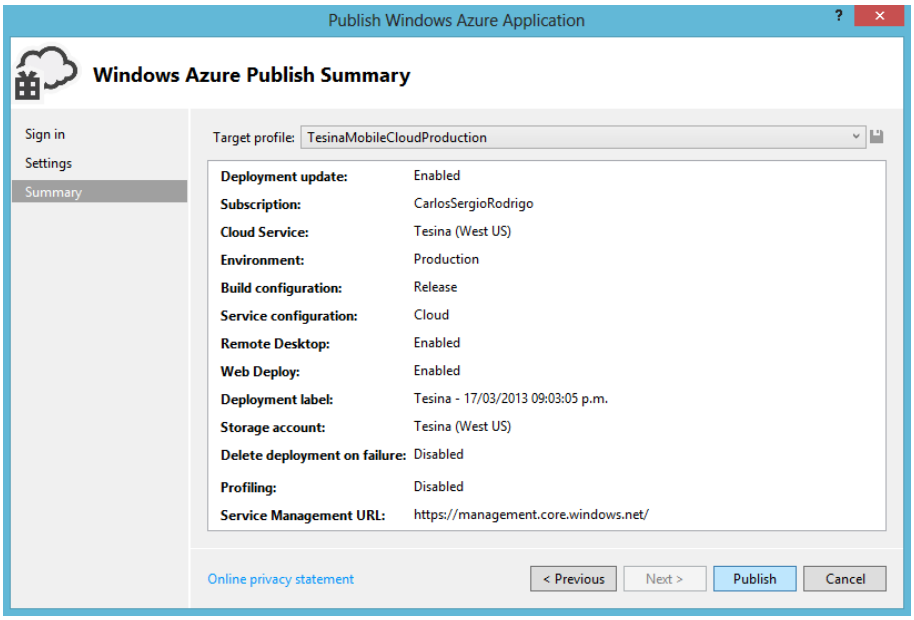

Figura 109 - Último paso para la publicación de los Roles de Windows Azure

5.3.1.3. Publicación de la aplicación Windows Phone

Para hacer que la aplicación sea pública para que los alumnos puedan descargarla en sus teléfonos con Windows Phone, la ésta debe estar disponible en el *Marketplace* de aplicaciones de esta plataforma. Para ello la aplicación sufre una evaluación y certificación antes de estar disponible para su descarga.

Para poder publicar la aplicación en el *Marketplace*, se debe contar primero con una cuenta de desarrollador. Esta cuenta tiene un costo de cien dólares anuales, y para solicitarla hace falta una cuenta Windows Live y una tarjeta de crédito para realizar el pago. Esta cuenta se gestiona en el sitio *Windows Phone Dev Center*[17].

Teniendo la cuenta habilitada para la publicación de aplicaciones, se debe acceder al sitio *Windows Phone Dev Center* utilizando la cuenta habilitada para la publicación de aplicaciones e ir al menu *Dashboard*. En esta pantalla seleccionar la opcion *SUBMIT APP*, como se muestra en la Figura 110.

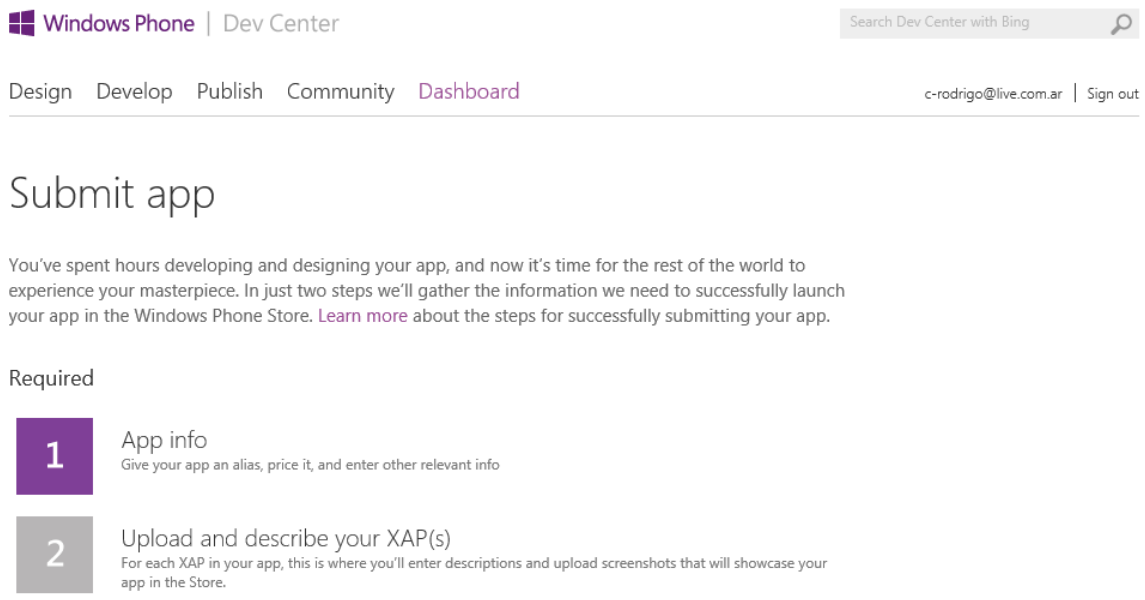

Figura 110 - Opción SUBMIT APP

[17] *Windows Phone Dev Center https://dev.windowsphone.com/en-us. Fecha de acceso: 18/03/2013*

Para publicar la aplicación se requieren dos pasos, detallar la información de la aplicación y proporcionar un archivo *.xap,* que es la aplicación Windows Phone compilada.

La información de la aplicación se provee seleccionando la opción *App info* que se muestra en la figura anterior, y se despliega el siguiente formulario:

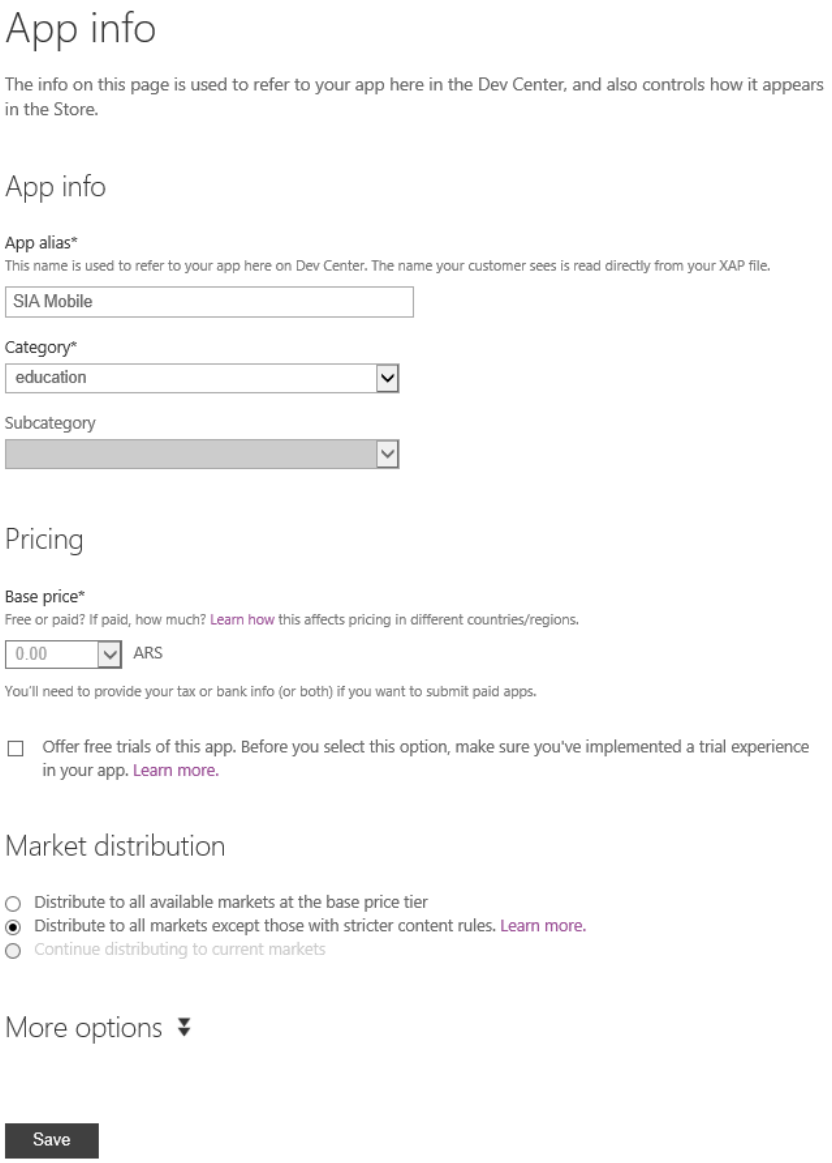

Figura 111 - Formulario de información de la aplicación a publicar

Luego de completados los campos que se muestran en la Figura 111, el proceso de publicación continua con la carga del archivo *.xap*. El mismo se encuentra en el directorio donde se depositan los archivos de la aplicación luego de ser compilada, por ejemplo *"Tesina\Tesina Source Code\TesinaMobileCloud\TesinaMobileCloud.Phone.Client\Bin\Release\TesinaMobileCloud.Phone.Client.xap".*

Luego de haber ubicado el archivo *.xap*, se debe seleccionar la opción "*Upload and describe your XAP(s)"* en la página para la carga de aplicaciones, descripta anteriormente, y proporcionar el archivo *.xap*.

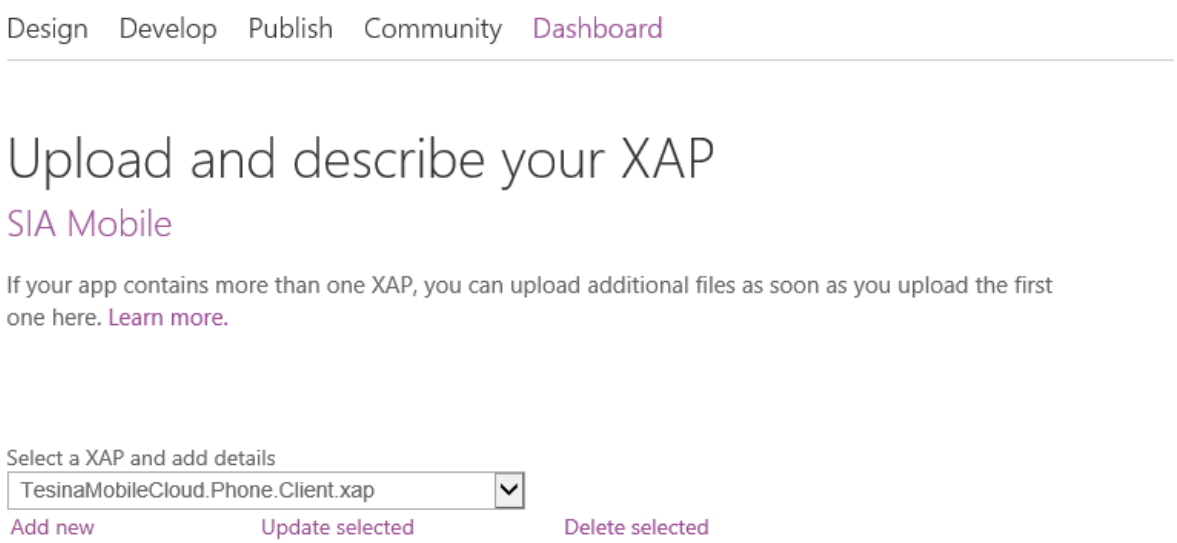

Figura 112 - Carga de archivo .xap

Para finalizar se debe presionar el botón *Submit,* luego de ello empieza el proceso de validación de la aplicación que si resulta satisfactorio da como resultado la publicación de la aplicación en el *Marketplace*. De modo contrario se envía un aviso por correo electrónico con el resultado de la validación y los aspectos que deben mejorarse.

6. Conclusiones

En esta sección se resumen las conclusiones del Trabajo Final de Carrera, teniendo en cuenta el cumplimiento de los objetivos definidos en la sección 3.1 Objetivo.

Se analiza cada objetivo específico, señalando si se cumplió parcial o totalmente, o si no se ha alcanzado.

Luego de analizar los objetivos específicos se procede a realizar el análisis del objetivo general.

6.1. Conclusiones respecto de los objetivos específicos

El primero de los objetivos específicos es **Utilizar una plataforma de Cloud Computing**. Se ha elegido el uso de Windows Azure, que es la plataforma de Cloud Computing de Microsoft. En la sección 4.2.1 Plataforma Windows Azure se describe la misma y las características que la componen.

Se ha construido parte de la solución que da como resultado el TFC sobre esta plataforma, como se indica en la sección 5.2.2.2 Desarrollo de la iteración. Al final del desarrollo en la sección 5.3.1 Entrega Final se ha realizado el despliegue de los componentes correspondientes a Windows Azure.

El siguiente objetivo específico es **Desarrollar un back-end de administración accesible vía web**. Para satisfacer este objetivo se creó un *WebRole* de WindowsAzure que es un sitio web de administración, como se describe en 5.2.2.2.2 AdminSite. El mismo permite crear, editar, eliminar, y listar carreras, materias, alumnos, relaciones entre alumnos y materias, y relaciones entre carreras y materias.

En cuanto al objetivo específico **Desarrollar un front-end para dispositivos móviles que es la tecnología más utilizada por los alumnos hoy en día**, se eligió la plataforma Windows Phone para desarrollar este *front-end*. Esta plataforma se describe en la sección 4.3 Windows Phone, y la construcción de la aplicación móvil se describe a lo largo de las cinco iteraciones que componen la sección 5.2 Construcción del software y posee la funcionalidades descriptas en 5.1.1 Historias de usuario.

Por último, el objetivo específico **Establecer una metodología para la construcción de la solución considerando prácticas de metodologías agiles**, se alcanza con la metodología propuesta en la sección 4.3 Metodología Utilizada. Que propone una metodología de carácter iterativo e incremental, influenciada por los principios establecidos en el Manifesto Ágil, descripto en la sección 4.2.1.

En base a lo desarrollado en los párrafos anteriores se puede concluir que los objetivos específicos se han cumplido satisfactoriamente.

6.2. Conclusiones respecto del objetivo general

El objetivo general y el objetivo principal han sido alcanzados mediante el estudio y el uso de las plataformas elegidas, Windows Phone y Windows Azure. La construcción de la solución de software obtenida ha permitido integrar los conocimientos adquiridos durante el estudio de las asignaturas nombradas en el objetivo general, así como también actualizarlos.

6.3. Conclusiones personales

A modo personal, el presente Trabajo Final de Carrera sirve para culminar mis estudios, pero a su vez lo considero como el punto de partida en una nueva etapa de mi carrera profesional. Los conocimientos adquiridos durante su elaboración me han servido para estudiar tecnologías y metodologías, que según mi impresión personal, ayudan a impulsar rápidamente una idea de negocio en el ámbito de las TICs. Es por ello que planeo continuar con su estudio para poder construir a partir de estas plataformas un emprendimiento personal.

6.4. Futuras Líneas de Investigación

Como se ha desarrollado en la sección 6.Conclusiones, la aplicación Mobile obtenida es una implementación particular para Windows Phone. Por lo que los futuros Trabajos Finales de Carrera podrían abordar desafíos como:

- Implementación de la Aplicación Mobile para Android
- Implementación de la Aplicación Mobile para iOS6
- Implementación de la Aplicación Mobile para Windows 8
- Implementación de la Aplicación Mobile para Windows Phone 8, utilizando las nuevas funcionalidades que esta actualización de la plataforma propone.
- Implementación de la Aplicación Mobile para BB10

Sería interesante también la investigación e implantación de soporte para redes sociales, como Twitter y Facebook, con funcionalidades como la de publicar el resultado obtenido en un examen o el cambio de la situación respecto a las asistencias en una materia.

También se puede agregar nuevas funcionalidades a la solución obtenida en el presente trabajo final de carrera, como ser:

Historia de usuario	*Novedades de la Universidad*	*Prioridad* **Media**
Como	***Estudiante***	*Tamaño* **10**
Quiero	Ver novedades de la Universidad.	
Entonces	Me mantendría informado acerca de las novedades sobre la Universidad y/o mi Facultad en particular.	

Criterios de aceptación	*Novedades de la Universidad*
Contexto	La universidad y/o la facultad a la que el alumno pertenece suelen publicar novedades y enviarlas por correo electrónico. Estas novedades pueden ser eventos sociales, congresos sobre temas de interés, entre otros.
Cuando	Al ingresar a la aplicación se debe visualizar una lista de las novedades

Resultado	Se visualiza la lista de novedades ordenadas por fecha de publicación y resaltadas las últimas añadidas.

Historia de usuario	*Aviso de novedades de la Universidad*	*Prioridad*	**Media**
Como	***Estudiante***	*Tamaño*	**5**
Quiero	Recibir aviso de novedades de la Universidad.		
Entonces	Me mantendría informado acerca de las novedades sobre la Universidad y/o mi Facultad en particular.		

Criterios de aceptación	*Aviso de novedades de la Universidad*
Contexto	Las novedades son las descriptas en la historia “Novedades de la Universidad”.
Cuando	Cuando se publique una novedad, se enviará una notificación informando que existen novedades nuevas.

Resultado	La notificación debe brindar información sobre el origen de la novedad, y permitirme dirigirme a ella.

Historia de usuario	*Información personal*	*Prioridad* **Media**
Como	***Estudiante***	*Tamaño* **5**
Quiero	Ver mi información personal y la relacionada las obligaciones académicas	
Entonces	Me mantendría informado sobre mi situación académica	

Criterios de aceptación	*Información personal*
Contexto	La información personal a mostrar es: • Cantidad de horas de congresos cumplidas • Año de cursada • Nombre y apellido • Próximo Examen • Matricula • Carrera

Cuando	Al ingresar a la aplicación se debe visualizar la información personal
Resultado	Puedo obtener una rápida información de mi información personal relacionada con mi situación académica

7. Bibliografía

Beck, K., & Andres, C. (2004). *Extreme Programming Explained Second Edition Embrace Change.* Addison-Wesley.

Britch, D., Cheung, F., Kinney, A., & Sharma, R. (2012). *Developing an Advanced Windows Phone 7.5 App that Connects to the Cloud.* Patterns & Practices, Microsoft.

Brown, C. (27 de Octubre de 2012). *User Story Template*. Obtenido de Better Projects: http://www.betterprojects.net/2011/03/user-story-template.html

Bus, S. (27 de Enero de 2012). *Windows Azure Service Bus*. Obtenido de Windows Azure: http://i.msdn.microsoft.com/gg318629.service-bus(es-es,MSDN.10).gif

Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Addison Wesley.

Control, A. (27 de Enero de 2012). *Windows Azure Access Control*. Obtenido de Windows Azure: http://i.msdn.microsoft.com/gg318629.access-control(es-es,MSDN.10).gif

Erich Gamma, R. H. (1994). *Design Patterns: Elements of Reusable Object Oriented Software.* Addison-Wesley.

Fowler Martin, B. K. (1999). *Refactoring: Improving the Design of Existing Code.* Addison Wesley.

Fowler, M. (13 de Noviembre de 2012). *P of EAA Catalog - Repository*. Obtenido de Martin Fowler: http://martinfowler.com/eaaCatalog/repository.html

Fowler, M. (16 de Febrero de 2013). *Data Transfer Object*. Obtenido de P of EAA Catalog: http://martinfowler.com/eaaCatalog/dataTransferObject.html

Galloway, J., Haack, P., Hanselman, S., Guthrie, S., & Conery, R. (2010). *Professional ASP.NET MVC 2.* Indianapolis, Indiana: Wiley Publishing,Inc.

GitHub. (18 de Marzo de 2013). *Home*. Obtenido de GitHub: https://github.com/

Ibrahim, E. (2009). *ASP.NET MVC 1.0 Test Driven Development.* Wrox.

Jurado, C. B. (2010). *Diseño Ágil con TDD.* iExpertos.

Krishnan, S. (2010). *Programming Windows Azure.* O'Reilly.

Lamb, S. (15 de Febrero de 2013). *Case Study: Using Windows Azure to create MicroFinance*. Obtenido de TechNet UK: http://blogs.technet.com/b/uktechnet/archive/2012/01/17/windows-azure-case-study-mando-group-microfinance-app.aspx

Manifesto, A. (18 de Octubre de 2012). *Agile Manifesto*. Obtenido de agilemanifesto: http://www.agilemanifesto.org/iso/es/

Manifesto, A. (18 de Octubre de 2012). *Principios del Manifiesto Ágil*. Obtenido de agilemanifesto: http://www.agilemanifesto.org/iso/es/principles.html

Microsoft Corporation. (06 de Octubre de 2012). *Central Application Hub with Home Page Menu (Panorama or Pivot Control) for Windows Phone*. Obtenido de Windows Phone Dev Center: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202892(v=vs.92).aspx

Microsoft Corporation. (24 de Septiembre de 2012). *Execution Model Overview for Windows Phone*. Obtenido de Windows Phone Dev Center: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008(v=vs.92).aspx

Microsoft Corporation. (08 de Octubre de 2012). *Push Notifications Overview*. Obtenido de Windows Phone Dev Center: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402558(v=vs.92).aspx

Microsoft Corporation. (15 de Febrero de 2013). *News360* . Obtenido de Microsoft Case Studies: http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?casestudyid=710000000283

Microsoft Corporation. (14 de Febrero de 2013). *T-Mobile USA* . Obtenido de Microsoft Case Studies: http://www.microsoft.com/casestudies/Windows-Azure/T-Mobile-USA/Mobile-Operator-Speeds-Time-to-Market-for-Innovative-Social-Networking-Solution/4000008598

Microsoft Corporation. (17 de Marzo de 2013). *Windows Azure*. Obtenido de Windows Azure: http://www.windowsazure.com

Microsoft Corporation. (17 de Marzo de 2013). *Windows Phone Dev Center*. Obtenido de Windows Phone Dev Center: https://dev.windowsphone.com/en-us

Smith, J. (16 de 02 de 2013). *WPF Apps With The Model-View-ViewModel Design Pattern*. Obtenido de MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx

Vaughan, D. (2012). *Windows Phone 7.5 Unleashed.* Sams.

Wikipedia. (4 de Julio de 2012). *Windows Mobile*. Obtenido de Wikipedia: http://es.wikipedia.org/wiki/Windows_Mobile

Windows Azure. (6 de Julio de 2012). *Downloads*. Obtenido de Windows Azure: http://www.windowsazure.com/en-us/downloads/?fb=es-es

8. Glosario

TFC: Trabajo Final de Carrera

WCF: *Windows Communication Foundation*, es la tecnología de Microsoft para el desarrollo de servicios para intercomunicar aplicaciones.

SDK: *Software Development Kit*. Conjunto de herramientas necesarias para el desarrollo de aplicaciones bajo una tecnología especifica.

REST: *Representational State Transfer*. Es una arquitectura para el diseño de aplicaciones en la red.

API: *Application Programming Interface*. Es una interfaz de programación que ofrece una aplicación para permitir la interacción con ella desde diferentes entornos.

Clave Primaria: en materia de base de datos representa una columna o dato que identifica unívocamente una fila dentro de una tabla.

Claves Foráneas: en materia de base de datos establece una relación de una tabla con otra tabla.

Marketplace: Tienda virtual donde se compran y descargan aplicaciones y/o juegos.

IsolateStorage: Es una sección de almacenamiento de datos dedicada a una aplicación y completamente aislada del resto.

Clase Abstracta: en el contexto de programación orientada a objetos, se refiere a una clase de la que no está permitido construir una instancia. Generalmente usada para definir un comportamiento de base que heredaran otras clases.

Test de unidad: es una prueba que tiene como objetivo comprobar que una sección de determinada del código fuente cumpla con la funcionalidad que dice que cumple.

Cobertura de Código: es una métrica en la construcción de software que indica cuando de nuestro código es evaluado por las pruebas de unidad.

Silverlight: tecnología de Microsoft para la construcción de aplicaciones ricas en contenido.

Windows Live: es una identidad virtual para el uso de los servicios que ofrece Microsoft.

HTTP: *Hypertext Transfer Protocol*. Protocolo de comunicación sin estado utilizado ampliamente en internet.

SQL: *Structured Query Language.* Es un lenguaje de alto para consulta y administración de datos sobre bases de datos relacionales.

XML: *Extensible Markup Language*. Es un lenguaje de marcas que permite estructurar información.

Backup: Copia de seguridad.

MPNS: *Microsoft Push Notification Service*. Es un servicio de Microsoft destinado a enviar notificaciones a distintos dispositivos móviles.

Downtime: Tiempo de caída de servicio.

Anexos

Índice

1. Anexo 1 – Iteración 1

SiaService

AzureLocalStorageTraceListener.cs

```
using System;
using System.Diagnostics;
using System.IO;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class AzureLocalStorageTraceListener : XmlWriterTraceListener
    {
        public AzureLocalStorageTraceListener()
            : base(Path.Combine(AzureLocalStorageTraceListener.GetLogDirectory().Path,
"SiaService.svclog"))
        {
        }

        public static DirectoryConfiguration GetLogDirectory()
        {
            var directory = new DirectoryConfiguration
                                        {
                                            Container = "wad-tracefiles",
                                            DirectoryQuotaInMB = 10,
                                            Path =

RoleEnvironment.GetLocalResource("SiaService.svclog").RootPath
                                        };
            return directory;
        }
    }
}
```

Global.asax.cs

```
using System;
using Ninject;
using Ninject.Web.Common;
using TesinaMobileCloud.Shared;

namespace SiaService
{
    public class Global : NinjectHttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            Application_Start();
        }

        protected override IKernel CreateKernel()
```

```
        {
            return new StandardKernel(new ServiceModule());
        }
    }
}
```

ISiaService.cs

```
using System.Collections.Generic;
using System.ServiceModel;
using System.ServiceModel.Web;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;

namespace SiaService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "ISiaService" in both code and config file together.
    [ServiceContract]
    public interface ISiaService
    {
        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate = "RetriveCoursesByCareer/{careerCode}")]
        IEnumerable<CourseDTO> RetriveCoursesByCareer(string careerCode);

        IList<Course> RetriveAllCourses();
    }
}
```

Service1.svc.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;

namespace SiaService
{
    public class SiaService : ISiaService
    {
        private readonly ICourseRepository _courseRepository;
        private readonly ICourseCareerRelationshipRepository
_courseCareerRelationshipRepository;

        public SiaService(ICourseRepository courseRepository,
ICourseCareerRelationshipRepository courseCareerRelationshipRepository)
        {
            _courseRepository = courseRepository;
            _courseCareerRelationshipRepository = courseCareerRelationshipRepository;
        }
```

```
        public IEnumerable<CourseDTO> RetriveCoursesByCareer(string careerCode)
        {
            var courses = _courseRepository.GetCoursesByCareer(careerCode);
            return courses.Select(course => new CourseDTO
                                        {
                                            CourseCode = course.CourseCode,
                                            Professor = course.Professor,
                                            Title = course.Title,
                                            YearOfCareer =
_courseCareerRelationshipRepository.GetCareerYearOfCourseByCareer(int.Parse(career
Code), course.CourseCode)
                                        });
        }

        public IList<Course> RetriveAllCourses()
        {
            return _courseRepository.GetAllCourses();
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // To enable the AzureLocalStorageTraceListner, uncomment relevent section in the
web.config
            DiagnosticMonitorConfiguration diagnosticConfig =
DiagnosticMonitor.GetDefaultInitialConfiguration();
            diagnosticConfig.Directories.ScheduledTransferPeriod =
TimeSpan.FromMinutes(1);

diagnosticConfig.Directories.DataSources.Add(AzureLocalStorageTraceListener.GetLogDir
ectory());

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

ServiceRoleTest

IServiceRoleFixture.cs

```
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using System.Collections.Generic;
using TesinaMobileCloud.Data;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Shared;

namespace ServiceRoleTest
{
   [TestClass]
   public class ServiceRoleFixture
   {
      const string CareerCode = "502";
      const int CourseCode = 120;

      private Mock<ICourseRepository> _repositoryCourse;
      private Mock<ICourseCareerRelationshipRepository> _repositoryRelationship;

      [TestInitialize]
      public void Setup()
      {
         _repositoryCourse = new Mock<ICourseRepository>();
         _repositoryRelationship = new Mock<ICourseCareerRelationshipRepository>();
         _repositoryCourse.Setup(m => m.GetCoursesByCareer(CareerCode)).Returns(new
List<Course>
                                                {
                                                   new Course(CourseCode,"Programación
I")
                                                   {
                                                      Professor = "Mg. Angeleri, Paula",
                                                   }
                                                });
         _repositoryRelationship.Setup(m =>
m.GetCareerYearOfCourseByCareer(int.Parse(CareerCode), CourseCode)).
            Returns(1);
      }


      /// <summary>
      /// Buscamos que devuelva una lista vacia de Cátedras. Diseñamos el metodo que
devulve la lista de Cátedras
      /// </summary>
      [TestMethod]
      public void RetriveCoursesByCareerCareerCodeEmptyList()
      {
         var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object);
```

```
            var result = sut.RetriveCoursesByCareer(CareerCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(IList<CourseDTO>));
        }

        /// <summary>
        /// Buscamos que devuelva una Cátedra. Diseñamos la entidad Cátedra.
        /// </summary>
        [TestMethod]
        public void RetriveCoursesByCareerCareerCodeOneCourse()
        {
            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object);

            var result = sut.RetriveCoursesByCareer(CareerCode).First();

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(CourseDTO));
            Assert.AreEqual(result.Title, "Programación I");
            Assert.AreEqual(result.Attendance, (int)AttendanceState.Danger);
            Assert.AreEqual(result.Professor, "Mg. Angeleri, Paula");
            Assert.AreEqual(result.CourseCode, CourseCode);
            Assert.AreEqual(result.YearOfCareer, 1);
        }

        [TestMethod]
        public void RetriveAllCourses()
        {
            _repositoryCourse.Setup(m => m.GetAllCourses()).Returns(new List<Course>
                                                {
                                                    new Course
                                                    {
                                                        CourseCode = 123,
                                                        Title = "Programación",
                                                        Professor = "Mg. Angeleri, Paula"
                                                    }
                                                });
            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object);

            var result = sut.RetriveAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(1, result.Count);
        }
    }
}
```

TesinaMobileCloud.Data

CloudStorageConfiguration.cs

```
using System;
```

```
using System.Configuration;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.Data
{
    public class CloudStorageConfiguration
    {
        public static CloudStorageAccount GetCloudAccount(string
cloudStorageConnectionStringName)
        {
            try
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
            catch (InvalidOperationException)
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
        }

        private static CloudStorageAccount SetConfigurationAndGetAccount(string
cloudStorageConnectionStringName)
        {
            SetConfigurationSettingsPublisher();
            return
CloudStorageAccount.FromConfigurationSetting(cloudStorageConnectionStringName);
        }

        private static void SetConfigurationSettingsPublisher()
        {
            CloudStorageAccount.SetConfigurationSettingPublisher((configName,
configSettingPublisher) =>
            {
                var configValue = ConfigurationManager.AppSettings[configName];
                if (RoleEnvironment.IsAvailable)
                    configValue = RoleEnvironment.GetConfigurationSettingValue(configName);
                configSettingPublisher(configValue);
            });
        }

    }
}
```

CourseDTO.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class CourseDTO
    {
        [DataMember]
```

```
        public int CourseCode { get; set; }
        [DataMember]
        public string Title { get; set; }
        [DataMember]
        public int Attendance { get; set; }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public int YearOfCareer { get; set; }
    }
}
```

Model

AzureEntity.cs

```
using System;
using System.Data.Services.Common;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataServiceEntity]
    [DataContract]
    public class AzureEntity
    {
        public string RowKey { get; set; }
        public string PartitionKey { get; set; }
        public DateTime Timestamp { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Career : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return RowKey; }
            set { RowKey = value; }
        }

        public Career()
```

```
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Career(int careerCode, string title)
        {
            CareerCode = careerCode;
            Title = title;
        }


    }
}
```

CareerCourseRelationship.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class CareerCourseRelationship : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public int YearInTheCareer { get; set; }

        public CareerCourseRelationship()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public CareerCourseRelationship(int careerCode, int courseCode)
        {
            CareerCode = careerCode;
            CourseCode = courseCode;
        }
    }
}
```

Course.cs

```
using System.Runtime.Serialization;
```

```
namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Course : AzureEntity
    {
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return string.IsNullOrEmpty(RowKey) ? string.Empty : RowKey; }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public string Professor { get; set; }

        public Course()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Course(int courseCode, string name)
        {
            CourseCode = courseCode;
            Title = name;
        }
    }
}
```

CareerCourseRelationshipRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CareerCourseRelationshipRepository :
ICourseCareerRelationshipRepository
    {
        private readonly ITable<CareerCourseRelationship> _table;

        public CareerCourseRelationshipRepository(ITable<CareerCourseRelationship> table)
        {
            _table = table;
        }

        public int GetCareerYearOfCourseByCareer(int careerCode, int courseCode)
        {
```

```
            return
                _table.GetSingleByCriteria(
                    relationship => relationship.CareerCode == careerCode &&
relationship.CourseCode == courseCode).
                    YearInTheCareer;
        }
    }
}
```

CourseRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CourseRepository : ICourseRepository
    {
        private readonly ITable<Course> _courseTable;
        private readonly ITable<CareerCourseRelationship> _careerCourseTable;

        public CourseRepository(ITable<Course> courseTable,
ITable<CareerCourseRelationship> careerCourseTable)
        {
            _courseTable = courseTable;
            _careerCourseTable = careerCourseTable;
        }

        public IList<Course> GetCoursesByCareer(string careerCode)
        {
            var courses = _careerCourseTable.GetByCriteria(c => c.CareerCode ==
int.Parse(careerCode));
            return courses.Select(careerCourseRelationship =>
_courseTable.GetSingleByCriteria(c => c.CourseCode ==
careerCourseRelationship.CourseCode)).ToList();
        }

        public IList<Course> GetAllCourses()
        {
            return _courseTable.GetByCriteria(c => !string.IsNullOrEmpty(c.PartitionKey));
        }

        public void AddCourse(Course course)
        {
            _courseTable.Add(course);
        }

        public void DeleteCourse(int partitionKey)
        {
            _courseTable.Delete(_courseTable.GetSingleByCriteria(c => c.CourseCode ==
partitionKey));
        }
```

```
    }
}
```

ICourseCareerRelationshipRepository.cs

```
namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseCareerRelationshipRepository
    {
        int GetCareerYearOfCourseByCareer(int careerCode, int courseCode);
    }
}
```

ICourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseRepository
    {
        IList<Course> GetCoursesByCareer(string careerCode);
        IList<Course> GetAllCourses();
        void AddCourse(Course course);
    }
}
```

AzureTable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;

namespace TesinaMobileCloud.Data.Table
{
    public class AzureTable<T> : ITable<T>
    {
        private readonly TableServiceContext _context;
        private readonly string _tableName;
        private IQueryable<T> Query { get; set; }

        public AzureTable(CloudStorageAccount cloudStorageAccount)
        {
            _tableName = typeof (T).Name;

            var table = new CloudTableClient(cloudStorageAccount.TableEndpoint.ToString(),
cloudStorageAccount.Credentials);
            _context = table.GetDataServiceContext();
            table.CreateTableIfNotExist(_tableName);
            Query = _context.CreateQuery<T>(_tableName).AsTableServiceQuery();
        }

        public IList<T> GetByCriteria(Func<T, bool> func)
        {
```

```
            return Query.Where(func).ToList();
        }

        public T GetSingleByCriteria(Func<T, bool> func)
        {
            return Query.SingleOrDefault(func);
        }

        public void Add(T entity)
        {
            _context.AddObject(_tableName, entity);
            _context.SaveChanges();
        }

        public void Delete(T entity)
        {
            _context.DeleteObject(entity);
            _context.SaveChanges();
        }
    }
}
```

ITable.cs

```
using System;
using System.Collections.Generic;

namespace TesinaMobileCloud.Data.Table
{
    public interface ITable<T>
    {
        IList<T> GetByCriteria(Func<T, bool> func);
        T GetSingleByCriteria(Func<T, bool> func);
        void Add(T entity);
        void Delete(T entity);
    }
}
```

TesinaMobileCloud.Data.Test

CourseCareerRelationshipRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class CourseCareerRelationshipRepositoryFixture
    {
        private Mock<ITable<CareerCourseRelationship>> _mockCourseCareerRelationship;
```

```
        [TestInitialize]
        public void Setup()
        {
            _mockCourseCareerRelationship = new
Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCareerYearOfCourseByCareer()
        {
            var career = 502;
            var course = 120;
            _mockCourseCareerRelationship.Setup(
                m => m.GetSingleByCriteria(It.IsAny<Func<CareerCourseRelationship,
bool>>())).Returns(new CareerCourseRelationship
                                                                    {
                                                                        CareerCode = career,
                                                                        CourseCode =
course,
                                                                        YearInTheCareer = 1
                                                                    });

            var sut = new
CareerCourseRelationshipRepository(_mockCourseCareerRelationship.Object);
            var result = sut.GetCareerYearOfCourseByCareer(career, course);

            Assert.AreEqual(1, result);


        }
    }
}
```

CourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class CourseRepositoryFixture
    {
        private Mock<ITable<Course>> _mockCourse;
        private Mock<ITable<CareerCourseRelationship>> _mockCareerCourse;

        [TestInitialize]
        public void Setup()
        {
```

```
            _mockCourse = new Mock<ITable<Course>>();
            _mockCareerCourse = new Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCoursesByCareer()
        {
            const int careerCode = 502;
            _mockCareerCourse.Setup(m =>
m.GetByCriteria(It.IsAny<Func<CareerCourseRelationship, bool>>())).Returns(new
List<CareerCourseRelationship>
                                                                        {
                                                                            new
CareerCourseRelationship(502, 123),
                                                                            new
CareerCourseRelationship(502, 124),
                                                                        });
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new Course(123, "Programación II"));
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetCoursesByCareer(careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 2);
        }

        [TestMethod]
        public void GetAllCourses()
        {
            _mockCourse.Setup(m => m.GetByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new List<Course>{ new Course(123, "Programación II")});
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 1);
        }

        [TestMethod]
        public void DeleteCourse()
        {
            var course = new Course(122, "Programación I");
            var courses = new List<Course>
                          {
                              new Course(123, "Programación II"),
                              course
                          };
            _mockCourse.Setup(m => m.Delete(course)).Callback(() =>
courses.Remove(course));
```

```
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
        bool>>())).Returns(course);

            var sut = new CourseRepository(_mockCourse.Object,
        _mockCareerCourse.Object);

            sut.DeleteCourse(122);

            Assert.AreEqual(1, courses.Count);
        }

        [TestMethod]
        public void AddCourse()
        {
            var course = new Course(123, "Programación II");
            _mockCourse.Setup(m => m.Add(It.IsAny<Course>())).Verifiable();

            var sut = new CourseRepository(_mockCourse.Object,
        _mockCareerCourse.Object);
            sut.AddCourse(course);
            _mockCourse.Verify();
        }
      }
}
```

TesinaMobileCloud.AdminSite

Global.asax.cs

```
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Web;
        using System.Web.Http;
        using System.Web.Mvc;
        using System.Web.Optimization;
        using System.Web.Routing;

        namespace TesinaMobileCloud.AdminSite
        {
            // Note: For instructions on enabling IIS6 or IIS7 classic mode,
            // visit http://go.microsoft.com/?LinkId=9394801

            public class MvcApplication : System.Web.HttpApplication
            {
                protected void Application_Start()
                {
                    AreaRegistration.RegisterAllAreas();

                    WebApiConfig.Register(GlobalConfiguration.Configuration);
                    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
                    RouteConfig.RegisterRoutes(RouteTable.Routes);
                    BundleConfig.RegisterBundles(BundleTable.Bundles);
                }
```

```
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.AdminSite
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

BundleConfig.cs

```
using System.Web;
using System.Web.Optimization;

namespace TesinaMobileCloud.AdminSite
{
    public class BundleConfig
    {
        // For more information on Bundling, visit
http://go.microsoft.com/fwlink/?LinkId=254725
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                        "~/Scripts/jquery-ui-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then,
when you're
            // ready for production, use the build tool at http://modernizr.com to pick only the
tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));
```

```
            bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

            bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
                        "~/Content/themes/base/jquery.ui.core.css",
                        "~/Content/themes/base/jquery.ui.resizable.css",
                        "~/Content/themes/base/jquery.ui.selectable.css",
                        "~/Content/themes/base/jquery.ui.accordion.css",
                        "~/Content/themes/base/jquery.ui.autocomplete.css",
                        "~/Content/themes/base/jquery.ui.button.css",
                        "~/Content/themes/base/jquery.ui.dialog.css",
                        "~/Content/themes/base/jquery.ui.slider.css",
                        "~/Content/themes/base/jquery.ui.tabs.css",
                        "~/Content/themes/base/jquery.ui.datepicker.css",
                        "~/Content/themes/base/jquery.ui.progressbar.css",
                        "~/Content/themes/base/jquery.ui.theme.css"));
        }
    }
}
```

FilterConfig.cs

```
using System.Web;
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

NinjectWebCommon.cs

```
using Ninject.Modules;
using TesinaMobileCloud.Shared;

[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NinjectWebCommon), "Start")]
[assembly:
WebActivator.ApplicationShutdownMethodAttribute(typeof(TesinaMobileCloud.AdminSite.A
pp_Start.NinjectWebCommon), "Stop")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
```

```
using Ninject.Web.Common;

public static class NinjectWebCommon
{
    private static readonly Bootstrapper bootstrapper = new Bootstrapper();

    /// <summary>
    /// Starts the application
    /// </summary>
    public static void Start()
    {
        DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
        DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
        bootstrapper.Initialize(CreateKernel);
    }

    /// <summary>
    /// Stops the application.
    /// </summary>
    public static void Stop()
    {
        bootstrapper.ShutDown();
    }

    /// <summary>
    /// Creates the kernel that will manage your application.
    /// </summary>
    /// <returns>The created kernel.</returns>
    private static IKernel CreateKernel()
    {
        var modules = new INinjectModule[] { new ServiceModule() };
        var kernel = new StandardKernel(modules);
        kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new Bootstrapper().Kernel);
        kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

        RegisterServices(kernel);
        return kernel;
    }

    /// <summary>
    /// Load your modules or register your services here!
    /// </summary>
    /// <param name="kernel">The kernel.</param>
    private static void RegisterServices(IKernel kernel)
    {
    }
}
```

RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

WebApiConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace TesinaMobileCloud.AdminSite
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Controllers

CareerController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
```

```
{
    public class CareerController : Controller
    {
        private readonly ITable<Career> _careers;

        public CareerController(ITable<Career> careers)
        {
            _careers = careers;
        }

//
        // GET: /Career/

        public ActionResult Index()
        {
            return View(_careers.GetByCriteria(career =>
!string.IsNullOrEmpty(career.PartitionKey)));
        }

        //
        // GET: /Career/Details/5

        public ActionResult Details(int id)
        {
            return View(_careers.GetSingleByCriteria(career => career.CareerCode == id));
        }

        //
        // GET: /Career/Create

        public ActionResult Create()
        {
            return View(new Career());
        }

        //
        // POST: /Career/Create

        [HttpPost]
        public ActionResult Create(Career career)
        {
            try
            {
                _careers.Add(career);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Edit/5
```

```
        public ActionResult Edit(int id)
        {
            return View();
        }

        //
        // POST: /Career/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var career = _careers.GetSingleByCriteria(career1 => career1.CareerCode ==
id);
                UpdateModel(career);
                _careers.Add(career);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Delete/5

        public ActionResult Delete(int id)
        {
            return View();
        }

        //
        // POST: /Career/Delete/5

        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                _careers.Delete(_careers.GetSingleByCriteria(c => c.CareerCode == id));

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CareerCourseRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
   public class CareerCourseRelationshipController : Controller
   {
      private readonly ITable<CareerCourseRelationship> _careerCourseRelationship;

      public CareerCourseRelationshipController(ITable<CareerCourseRelationship>
careerCourseRelationship)
      {
         _careerCourseRelationship = careerCourseRelationship;
      }

//
      // GET: /CareerCourseRelationship/

      public ActionResult Index()
      {
         return View(_careerCourseRelationship.GetByCriteria(relationship =>
relationship.CareerCode != 0 ));
      }

      //
      // GET: /CareerCourseRelationship/Details/5

      public ActionResult Details(int id)
      {
         return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CareerCode == id ));
      }

      //
      // GET: /CareerCourseRelationship/Create

      public ActionResult Create()
      {
         return View(new CareerCourseRelationship());
      }

      //
      // POST: /CareerCourseRelationship/Create

      [HttpPost]
      public ActionResult Create(CareerCourseRelationship careerCourseRelationship)
```

```
{
    try
    {
        _careerCourseRelationship.Add(careerCourseRelationship);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /CareerCourseRelationship/Edit/5

public ActionResult Edit(int id)
{
    return View();
}

//
// POST: /CareerCourseRelationship/Edit/5

[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
    try
    {
        // TODO: Add update logic here

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /CareerCourseRelationship/Delete/5

public ActionResult Delete(int id)
{
    return View();
}

//
// POST: /CareerCourseRelationship/Delete/5

[HttpPost]
public ActionResult Delete(int id, FormCollection collection)
{
    try
    {
```

```
                // TODO: Add delete logic here

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CourseController.cs

```
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CourseController : Controller
    {
        private readonly ITable<Course> _courses;

        public CourseController(ITable<Course> courses)
        {
            _courses = courses;
        }

        public ActionResult Index()
        {
            return View(_courses.GetByCriteria(course =>
!string.IsNullOrEmpty(course.PartitionKey)));
        }

        //
        // GET: /Course/Details/5

        public ActionResult Details(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.CourseCode == id));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new Course(0, string.Empty));
        }

        //
        // POST: /Course/Create
```

```
        [HttpPost]
        public ActionResult Create(Course course)
        {
            try
            {
                _courses.Add(course);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.CourseCode == id));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _courses.GetSingleByCriteria(course =>
course.CourseCode == id);
                UpdateModel(courseInCloud);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
                _courses.Delete( _courses.GetSingleByCriteria(c => c.PartitionKey == id &&
c.RowKey == row));
                return RedirectToAction("Index");
            }
            catch
```

```
            {
                return View();
            }
        }

        //
        // POST: /Course/Delete/5


    }
}
```

HomeController.cs

```
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC
application.";

            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

Views

_ViewStart.cshtml

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Career

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Career
```

```
@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Career</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CareerCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>
```

```
<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.CareerCode)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Title)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.CareerCode }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Career</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CareerCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
```

```
        </div>

        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Career>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CareerCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Title)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id = item.CareerCode }) |
            @Html.ActionLink("Details", "Details", new { id = item.CareerCode }) |
            @Html.ActionLink("Delete", "Delete", new { id = item.CareerCode })
        </td>
```

```
    </tr>
}

</table>
```

CareerCourseRelationship

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>CareerCourseRelationship</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CareerCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.CourseCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.YearInTheCareer)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>


        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}
```

```
<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.CareerCourseRelationship>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CareerCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.CourseCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.YearInTheCareer)
        </th>

        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CourseCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.YearInTheCareer)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
        </td>
    </tr>
```

```
}

</table>
```

Course

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Course</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CourseCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Professor)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<fieldset>
    <legend>Course</legend>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.CourseCode)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Title)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Professor)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Delete" /> |
        @Html.ActionLink("Back to List", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
```

```
}

<h2>Details</h2>

<fieldset>
    <legend>Course</legend>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.CourseCode)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Title)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
    <div class="display-label">
        @Html.DisplayNameFor(model => model.Professor)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.CourseCode }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Course</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CourseCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
```

```
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Professor)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>
        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Course>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CourseCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Title)
        </th>
```

```
            <th>
                @Html.DisplayNameFor(model => model.Professor)
            </th>
            <th></th>
        </tr>

    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.CourseCode)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Professor)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id=item.CourseCode }) |
                @Html.ActionLink("Details", "Details", new { id=item.CourseCode }) |
                @Html.ActionLink("Delete", "Delete", new { id=item.CourseCode, row = item.Title })
            </td>
        </tr>
    }

    </table>
```

Home

Index.cshtml

```
    @{
        ViewBag.Title = "Sitio de Administración - Tesina Carlos Rodrigo";
    }
    @section featured {
        <section class="featured">
            <div class="content-wrapper">
                <hgroup class="title">
                    <h1>@ViewBag.Title.</h1>
                    <h2>@ViewBag.Message</h2>
                </hgroup>
                <p>
                    Este sitio provee las funciones de administración y carga de datos necesarios
    para el desarrollo de la tesina.
                    La necesidad surge de simular los sistemas de asistencias y administración de
    los alumnos, para solo dedicarse a
                    la implementación del cliente Mobile.
                </p>
            </div>
        </section>
    }
```

Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<hgroup class="title">
    <h1 class="error">Error.</h1>
    <h2 class="error">An error occurred while processing your request.</h2>
</hgroup>
```

Shared

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title - My ASP.NET MVC Application</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width" />
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
                </div>
                <div class="float-right">
                    <section id="login">
                        @Html.Partial("_LoginPartial")
                    </section>
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("Cátedras", "Index", "Course")</li>
                            <li>@Html.ActionLink("Carreras", "Index", "Career")</li>
                            <li>@Html.ActionLink("Cátedras-Carreras", "Index",
"CareerCourseRelationship")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
```

```
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
            </div>
        </footer>

        @Scripts.Render("~/bundles/jquery")
        @RenderSection("scripts", required: false)
    </body>
</html>
```

TesinaMobileCloud.Shared

ServiceModule.cs

```
using Microsoft.WindowsAzure;
using Ninject.Modules;
using TesinaMobileCloud.Data;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Shared
{
    public class ServiceModule : NinjectModule
    {
        public override void Load()
        {
            Bind<CloudStorageAccount>().ToMethod(context =>
CloudStorageConfiguration.GetCloudAccount("DataConnectionString"));

            Bind<ITable<Course>>().To<AzureTable<Course>>();
            Bind<ITable<Career>>().To<AzureTable<Career>>();

Bind<ITable<CareerCourseRelationship>>().To<AzureTable<CareerCourseRelationship>>(
);

            Bind<ICourseRepository>().To<CourseRepository>();

Bind<ICourseCareerRelationshipRepository>().To<CareerCourseRelationshipRepository>()
;
        }
    }
}
```

2. Anexo 2 – Iteración 2

2.1 SiaService

AzureLocalStorageTraceListener.cs

```
using System;
using System.Diagnostics;
using System.IO;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
   public class AzureLocalStorageTraceListener : XmlWriterTraceListener
   {
      public AzureLocalStorageTraceListener()
         : base(Path.Combine(AzureLocalStorageTraceListener.GetLogDirectory().Path,
"SiaService.svclog"))
      {
      }

      public static DirectoryConfiguration GetLogDirectory()
      {
         var directory = new DirectoryConfiguration
                                    {
                                       Container = "wad-tracefiles",
                                       DirectoryQuotaInMB = 10,
                                       Path =

RoleEnvironment.GetLocalResource("SiaService.svclog").RootPath
                                    };
         return directory;
      }
   }
}
```

Global.asax.cs

```
using System;
using Ninject;
using Ninject.Web.Common;
using TesinaMobileCloud.Shared;

namespace SiaService
{
   public class Global : NinjectHttpApplication
   {
      protected void Application_Start(object sender, EventArgs e)
      {
         Application_Start();
      }

      protected override IKernel CreateKernel()
```

```
        {
            return new StandardKernel(new ServiceModule());
        }
    }
}
```

ISiaService.cs

```
using System.Collections.Generic;
using System.ServiceModel;
using System.ServiceModel.Web;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;

namespace SiaService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "ISiaService" in both code and config file together.
    [ServiceContract]
    public interface ISiaService
    {
        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate = "RetriveCoursesByCareer/{careerCode}")]
        IEnumerable<CourseDTO> RetriveCoursesByCareer(string careerCode);

        IList<Course> RetriveAllCourses();
    }
}
```

Service1.svc.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;

namespace SiaService
{
    public class SiaService : ISiaService
    {
        private readonly ICourseRepository _courseRepository;
        private readonly ICourseCareerRelationshipRepository
_courseCareerRelationshipRepository;

        public SiaService(ICourseRepository courseRepository,
ICourseCareerRelationshipRepository courseCareerRelationshipRepository)
        {
            _courseRepository = courseRepository;
            _courseCareerRelationshipRepository = courseCareerRelationshipRepository;
```

```
        }

        public IEnumerable<CourseDTO> RetriveCoursesByCareer(string careerCode)
        {
            var courses = _courseRepository.GetCoursesByCareer(careerCode);
            return courses.Select(course => new CourseDTO
                                        {
                                            CourseCode = course.CourseCode,
                                            Professor = course.Professor,
                                            Title = course.Title,
                                            YearOfCareer =
_courseCareerRelationshipRepository.GetCareerYearOfCourseByCareer(int.Parse(career
Code), course.CourseCode)
                                        });
        }

        public IList<Course> RetriveAllCourses()
        {
            return _courseRepository.GetAllCourses();
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // To enable the AzureLocalStorageTraceListner, uncomment relevent section in the
web.config
            DiagnosticMonitorConfiguration diagnosticConfig =
DiagnosticMonitor.GetDefaultInitialConfiguration();
            diagnosticConfig.Directories.ScheduledTransferPeriod =
TimeSpan.FromMinutes(1);

diagnosticConfig.Directories.DataSources.Add(AzureLocalStorageTraceListener.GetLogDir
ectory());

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
```

```
}
```

ServiceRoleTest

IServiceRoleFixture.cs

```
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using System.Collections.Generic;
using TesinaMobileCloud.Data;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Shared;

namespace ServiceRoleTest
{
   [TestClass]
   public class ServiceRoleFixture
   {
      const string CareerCode = "502";
      const int CourseCode = 120;

      private Mock<ICourseRepository> _repositoryCourse;
      private Mock<ICourseCareerRelationshipRepository> _repositoryRelationship;

      [TestInitialize]
      public void Setup()
      {
         _repositoryCourse = new Mock<ICourseRepository>();
         _repositoryRelationship = new Mock<ICourseCareerRelationshipRepository>();
         _repositoryCourse.Setup(m => m.GetCoursesByCareer(CareerCode)).Returns(new
List<Course>
                                                {
                                                   new Course(CourseCode,"Programación
I")
                                                   {
                                                      Professor = "Mg. Angeleri, Paula",
                                                   }
                                                });
         _repositoryRelationship.Setup(m =>
m.GetCareerYearOfCourseByCareer(int.Parse(CareerCode), CourseCode)).
             Returns(1);
      }


      /// <summary>
      /// Buscamos que devuelva una lista vacia de Cátedras. Diseñamos el metodo que
devulve la lista de Cátedras
      /// </summary>
      [TestMethod]
      public void RetriveCoursesByCareerCareerCodeEmptyList()
      {
```

```
            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object);

            var result = sut.RetriveCoursesByCareer(CareerCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(IList<CourseDTO>));
        }

        /// <summary>
        /// Buscamos que devuelva una Cátedra. Diseñamos la entidad Cátedra.
        /// </summary>
        [TestMethod]
        public void RetriveCoursesByCareerCareerCodeOneCourse()
        {
            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object);

            var result = sut.RetriveCoursesByCareer(CareerCode).First();

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(CourseDTO));
            Assert.AreEqual(result.Title, "Programación I");
            Assert.AreEqual(result.Attendance, (int)AttendanceState.Danger);
            Assert.AreEqual(result.Professor, "Mg. Angeleri, Paula");
            Assert.AreEqual(result.CourseCode, CourseCode);
            Assert.AreEqual(result.YearOfCareer, 1);
        }

        [TestMethod]
        public void RetriveAllCourses()
        {
            _repositoryCourse.Setup(m => m.GetAllCourses()).Returns(new List<Course>
                                                {
                                                    new Course
                                                    {
                                                        CourseCode = 123,
                                                        Title = "Programación",
                                                        Professor = "Mg. Angeleri, Paula"
                                                    }
                                                });
            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object);

            var result = sut.RetriveAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(1, result.Count);
        }
    }
}
```

TesinaMobileCloud.Data

CloudStorageConfiguration.cs

```
using System;
using System.Configuration;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.Data
{
    public class CloudStorageConfiguration
    {
        public static CloudStorageAccount GetCloudAccount(string
cloudStorageConnectionStringName)
        {
            try
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
            catch (InvalidOperationException)
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
        }

        private static CloudStorageAccount SetConfigurationAndGetAccount(string
cloudStorageConnectionStringName)
        {
            SetConfigurationSettingsPublisher();
            return
CloudStorageAccount.FromConfigurationSetting(cloudStorageConnectionStringName);
        }

        private static void SetConfigurationSettingsPublisher()
        {
            CloudStorageAccount.SetConfigurationSettingPublisher((configName,
configSettingPublisher) =>
            {
                var configValue = ConfigurationManager.AppSettings[configName];
                if (RoleEnvironment.IsAvailable)
                    configValue = RoleEnvironment.GetConfigurationSettingValue(configName);
                configSettingPublisher(configValue);
            });
        }

    }
}
```

CourseDTO.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
```

```
    [DataContract]
    public class CourseDTO
    {
        [DataMember]
        public int CourseCode { get; set; }
        [DataMember]
        public string Title { get; set; }
        [DataMember]
        public int Attendance { get; set; }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public int YearOfCareer { get; set; }
    }
}
```

Model

AzureEntity.cs

```
using System;
using System.Data.Services.Common;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataServiceEntity]
    [DataContract]
    public class AzureEntity
    {
        public string RowKey { get; set; }
        public string PartitionKey { get; set; }
        public DateTime Timestamp { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Career : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return RowKey; }
```

```
            set { RowKey = value; }
        }

        public Career()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Career(int careerCode, string title)
        {
            CareerCode = careerCode;
            Title = title;
        }


    }
}
```

CareerCourseRelationship.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class CareerCourseRelationship : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public int YearInTheCareer { get; set; }

        public CareerCourseRelationship()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public CareerCourseRelationship(int careerCode, int courseCode)
        {
            CareerCode = careerCode;
            CourseCode = courseCode;
        }
    }
```

```
}
```

Course.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
   [DataContract]
   public class Course : AzureEntity
   {
      [DataMember]
      public int CourseCode
      {
         get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
         set { PartitionKey = value.ToString(); }
      }
      [DataMember]
      public string Title
      {
         get { return string.IsNullOrEmpty(RowKey) ? string.Empty : RowKey; }
         set { RowKey = value.ToString(); }
      }
      [DataMember]
      public string Professor { get; set; }

      public Course()
      {
         PartitionKey = string.Empty;
         RowKey = string.Empty;
      }

      public Course(int courseCode, string name)
      {
         CourseCode = courseCode;
         Title = name;
      }
   }
}
```

CareerCourseRelationshipRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
   public class CareerCourseRelationshipRepository :
ICourseCareerRelationshipRepository
   {
      private readonly ITable<CareerCourseRelationship> _table;

      public CareerCourseRelationshipRepository(ITable<CareerCourseRelationship> table)
      {
         _table = table;
```

```
        }

        public int GetCareerYearOfCourseByCareer(int careerCode, int courseCode)
        {
            return
                _table.GetSingleByCriteria(
                    relationship => relationship.CareerCode == careerCode &&
relationship.CourseCode == courseCode).
                    YearInTheCareer;
        }
    }
}
```

CourseRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CourseRepository : ICourseRepository
    {
        private readonly ITable<Course> _courseTable;
        private readonly ITable<CareerCourseRelationship> _careerCourseTable;

        public CourseRepository(ITable<Course> courseTable,
ITable<CareerCourseRelationship> careerCourseTable)
        {
            _courseTable = courseTable;
            _careerCourseTable = careerCourseTable;
        }

        public IList<Course> GetCoursesByCareer(string careerCode)
        {
            var courses = _careerCourseTable.GetByCriteria(c => c.CareerCode ==
int.Parse(careerCode));
            return courses.Select(careerCourseRelationship =>
_courseTable.GetSingleByCriteria(c => c.CourseCode ==
careerCourseRelationship.CourseCode)).ToList();
        }

        public IList<Course> GetAllCourses()
        {
            return _courseTable.GetByCriteria(c => !string.IsNullOrEmpty(c.PartitionKey));
        }

        public void AddCourse(Course course)
        {
            _courseTable.Add(course);
        }

        public void DeleteCourse(int partitionKey)
```

```
            {
                _courseTable.Delete(_courseTable.GetSingleByCriteria(c => c.CourseCode ==
        partitionKey));
            }
        }
    }
```

ICourseCareerRelationshipRepository.cs

```
    namespace TesinaMobileCloud.Data.Repository.Interfaces
    {
        public interface ICourseCareerRelationshipRepository
        {
            int GetCareerYearOfCourseByCareer(int careerCode, int courseCode);
        }
    }
```

ICourseRepository.cs

```
    using System.Collections.Generic;
    using TesinaMobileCloud.Data.Model;

    namespace TesinaMobileCloud.Data.Repository.Interfaces
    {
        public interface ICourseRepository
        {
            IList<Course> GetCoursesByCareer(string careerCode);
            IList<Course> GetAllCourses();
            void AddCourse(Course course);
        }
    }
```

AzureTable.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;

    namespace TesinaMobileCloud.Data.Table
    {
        public class AzureTable<T> : ITable<T>
        {
            private readonly TableServiceContext _context;
            private readonly string _tableName;
            private IQueryable<T> Query { get; set; }

            public AzureTable(CloudStorageAccount cloudStorageAccount)
            {
                _tableName = typeof (T).Name;

                var table = new CloudTableClient(cloudStorageAccount.TableEndpoint.ToString(),
        cloudStorageAccount.Credentials);
                _context = table.GetDataServiceContext();
                table.CreateTableIfNotExist(_tableName);
                Query = _context.CreateQuery<T>(_tableName).AsTableServiceQuery();
```

```
        }

        public IList<T> GetByCriteria(Func<T, bool> func)
        {
            return Query.Where(func).ToList();
        }

        public T GetSingleByCriteria(Func<T, bool> func)
        {
            return Query.SingleOrDefault(func);
        }

        public void Add(T entity)
        {
            _context.AddObject(_tableName, entity);
            _context.SaveChanges();
        }

        public void Delete(T entity)
        {
            _context.DeleteObject(entity);
            _context.SaveChanges();
        }
    }
}
```

ITable.cs

```
using System;
using System.Collections.Generic;

namespace TesinaMobileCloud.Data.Table
{
    public interface ITable<T>
    {
        IList<T> GetByCriteria(Func<T, bool> func);
        T GetSingleByCriteria(Func<T, bool> func);
        void Add(T entity);
        void Delete(T entity);
    }
}
```

TesinaMobileCloud.Data.Test

CourseCareerRelationshipRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
```

```
    public class CourseCareerRelationshipRepositoryFixture
    {
        private Mock<ITable<CareerCourseRelationship>> _mockCourseCareerRelationship;

        [TestInitialize]
        public void Setup()
        {
            _mockCourseCareerRelationship = new
Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCareerYearOfCourseByCareer()
        {
            var career = 502;
            var course = 120;
            _mockCourseCareerRelationship.Setup(
                m => m.GetSingleByCriteria(It.IsAny<Func<CareerCourseRelationship,
bool>>())).Returns(new CareerCourseRelationship
                                                                        {
                                                                            CareerCode = career,
                                                                            CourseCode =
course,
                                                                            YearInTheCareer = 1
                                                                        });

            var sut = new
CareerCourseRelationshipRepository(_mockCourseCareerRelationship.Object);
            var result = sut.GetCareerYearOfCourseByCareer(career, course);

            Assert.AreEqual(1, result);


        }
    }
}
```

CourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class CourseRepositoryFixture
    {
        private Mock<ITable<Course>> _mockCourse;
        private Mock<ITable<CareerCourseRelationship>> _mockCareerCourse;
```

```
        [TestInitialize]
        public void Setup()
        {
            _mockCourse = new Mock<ITable<Course>>();
            _mockCareerCourse = new Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCoursesByCareer()
        {
            const int careerCode = 502;
            _mockCareerCourse.Setup(m =>
m.GetByCriteria(It.IsAny<Func<CareerCourseRelationship, bool>>())).Returns(new
List<CareerCourseRelationship>
                                                                        {
                                                                            new
CareerCourseRelationship(502, 123),
                                                                            new
CareerCourseRelationship(502, 124),
                                                                        });
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new Course(123, "Programación II"));
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetCoursesByCareer(careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 2);
        }

        [TestMethod]
        public void GetAllCourses()
        {
            _mockCourse.Setup(m => m.GetByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new List<Course>{ new Course(123, "Programación II")});
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 1);
        }

        [TestMethod]
        public void DeleteCourse()
        {
            var course = new Course(122, "Programación I");
            var courses = new List<Course>
                        {
                            new Course(123, "Programación II"),
                            course
```

```
                        };
                _mockCourse.Setup(m => m.Delete(course)).Callback(() =>
        courses.Remove(course));
                _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
        bool>>())).Returns(course);

                var sut = new CourseRepository(_mockCourse.Object,
        _mockCareerCourse.Object);

                sut.DeleteCourse(122);

                Assert.AreEqual(1, courses.Count);
            }

            [TestMethod]
            public void AddCourse()
            {
                var course = new Course(123, "Programación II");
                _mockCourse.Setup(m => m.Add(It.IsAny<Course>())).Verifiable();

                var sut = new CourseRepository(_mockCourse.Object,
        _mockCareerCourse.Object);
                sut.AddCourse(course);
                _mockCourse.Verify();
            }
        }
}
```

TesinaMobileCloud.AdminSite

Global.asax.cs

```
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Web;
        using System.Web.Http;
        using System.Web.Mvc;
        using System.Web.Optimization;
        using System.Web.Routing;

        namespace TesinaMobileCloud.AdminSite
        {
            // Note: For instructions on enabling IIS6 or IIS7 classic mode,
            // visit http://go.microsoft.com/?LinkId=9394801

            public class MvcApplication : System.Web.HttpApplication
            {
                protected void Application_Start()
                {
                    AreaRegistration.RegisterAllAreas();

                    WebApiConfig.Register(GlobalConfiguration.Configuration);
                    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
```

```
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.AdminSite
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

BundleConfig.cs

```
using System.Web;
using System.Web.Optimization;

namespace TesinaMobileCloud.AdminSite
{
    public class BundleConfig
    {
        // For more information on Bundling, visit
http://go.microsoft.com/fwlink/?LinkId=254725
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                        "~/Scripts/jquery-ui-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then,
when you're
```

```
            // ready for production, use the build tool at http://modernizr.com to pick only the
        tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                    "~/Scripts/modernizr-*"));

            bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

            bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
                    "~/Content/themes/base/jquery.ui.core.css",
                    "~/Content/themes/base/jquery.ui.resizable.css",
                    "~/Content/themes/base/jquery.ui.selectable.css",
                    "~/Content/themes/base/jquery.ui.accordion.css",
                    "~/Content/themes/base/jquery.ui.autocomplete.css",
                    "~/Content/themes/base/jquery.ui.button.css",
                    "~/Content/themes/base/jquery.ui.dialog.css",
                    "~/Content/themes/base/jquery.ui.slider.css",
                    "~/Content/themes/base/jquery.ui.tabs.css",
                    "~/Content/themes/base/jquery.ui.datepicker.css",
                    "~/Content/themes/base/jquery.ui.progressbar.css",
                    "~/Content/themes/base/jquery.ui.theme.css"));
        }
    }
}
```

FilterConfig.cs

```
using System.Web;
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

NinjectWebCommon.cs

```
using Ninject.Modules;
using TesinaMobileCloud.Shared;

[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NinjectWebCommon), "Start")]
[assembly:
WebActivator.ApplicationShutdownMethodAttribute(typeof(TesinaMobileCloud.AdminSite.A
pp_Start.NinjectWebCommon), "Stop")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System;
    using System.Web;
```

```
    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;

    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var modules = new INinjectModule[] { new ServiceModule() };
            var kernel = new StandardKernel(modules);
            kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new Bootstrapper().Kernel);
            kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

            RegisterServices(kernel);
            return kernel;
        }

        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
}
```

RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

WebApiConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace TesinaMobileCloud.AdminSite
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Controllers

CareerController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerController : Controller
    {
        private readonly ITable<Career> _careers;

        public CareerController(ITable<Career> careers)
        {
            _careers = careers;
        }

//
        // GET: /Career/

        public ActionResult Index()
        {
            return View(_careers.GetByCriteria(career =>
!string.IsNullOrEmpty(career.PartitionKey)));
        }

        //
        // GET: /Career/Details/5

        public ActionResult Details(int id)
        {
            return View(_careers.GetSingleByCriteria(career => career.CareerCode == id));
        }

        //
        // GET: /Career/Create

        public ActionResult Create()
        {
            return View(new Career());
        }

        //
        // POST: /Career/Create

        [HttpPost]
        public ActionResult Create(Career career)
        {
            try
            {
                _careers.Add(career);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
```

```
        }

        //
        // GET: /Career/Edit/5

        public ActionResult Edit(int id)
        {
            return View();
        }

        //
        // POST: /Career/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var career = _careers.GetSingleByCriteria(career1 => career1.CareerCode ==
id);
                UpdateModel(career);
                _careers.Add(career);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Delete/5

        public ActionResult Delete(int id)
        {
            return View();
        }

        //
        // POST: /Career/Delete/5

        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                _careers.Delete(_careers.GetSingleByCriteria(c => c.CareerCode == id));

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
```

```
            }
        }
    }
}
```

CareerCourseRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerCourseRelationshipController : Controller
    {
        private readonly ITable<CareerCourseRelationship> _careerCourseRelationship;

        public CareerCourseRelationshipController(ITable<CareerCourseRelationship>
careerCourseRelationship)
        {
            _careerCourseRelationship = careerCourseRelationship;
        }

//
        // GET: /CareerCourseRelationship/

        public ActionResult Index()
        {
            return View(_careerCourseRelationship.GetByCriteria(relationship =>
relationship.CareerCode != 0 ));
        }

        //
        // GET: /CareerCourseRelationship/Details/5

        public ActionResult Details(int id)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CareerCode == id ));
        }

        //
        // GET: /CareerCourseRelationship/Create

        public ActionResult Create()
        {
            return View(new CareerCourseRelationship());
        }

        //
```

```
// POST: /CareerCourseRelationship/Create

[HttpPost]
public ActionResult Create(CareerCourseRelationship careerCourseRelationship)
{
    try
    {
        _careerCourseRelationship.Add(careerCourseRelationship);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /CareerCourseRelationship/Edit/5

public ActionResult Edit(int id)
{
    return View();
}

//
// POST: /CareerCourseRelationship/Edit/5

[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
    try
    {
        // TODO: Add update logic here

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /CareerCourseRelationship/Delete/5

public ActionResult Delete(int id)
{
    return View();
}

//
// POST: /CareerCourseRelationship/Delete/5

[HttpPost]
```

```
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                // TODO: Add delete logic here

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CourseController.cs

```
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CourseController : Controller
    {
        private readonly ITable<Course> _courses;

        public CourseController(ITable<Course> courses)
        {
            _courses = courses;
        }

        public ActionResult Index()
        {
            return View(_courses.GetByCriteria(course =>
!string.IsNullOrEmpty(course.PartitionKey)));
        }

        //
        // GET: /Course/Details/5

        public ActionResult Details(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.CourseCode == id));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new Course(0, string.Empty));
        }
```

```
        //
        // POST: /Course/Create

        [HttpPost]
        public ActionResult Create(Course course)
        {
            try
            {
                _courses.Add(course);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.CourseCode == id));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _courses.GetSingleByCriteria(course =>
course.CourseCode == id);
                UpdateModel(courseInCloud);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
```

```
                _courses.Delete( _courses.GetSingleByCriteria(c => c.PartitionKey == id &&
c.RowKey == row));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // POST: /Course/Delete/5


    }
}
```

HomeController.cs

```
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC
application.";

            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

Views

_ViewStart.cshtml

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
```

```
}
```

Career

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Career</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CareerCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Career
```

```
@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.CareerCode)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Title)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.CareerCode }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Career</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.CareerCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>
```

```
        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Career>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CareerCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Title)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
```

```
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.CareerCode }) |
                @Html.ActionLink("Details", "Details", new { id = item.CareerCode }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.CareerCode })
            </td>
        </tr>
    }

    </table>
```

CareerCourseRelationship

Create.cshtml

```
    @model TesinaMobileCloud.Data.Model.CareerCourseRelationship

    @{
        ViewBag.Title = "Create";
        Layout = "~/Views/Shared/_Layout.cshtml";
    }

    <h2>Create</h2>

    @using (Html.BeginForm()) {
        @Html.ValidationSummary(true)

        <fieldset>
            <legend>CareerCourseRelationship</legend>

            <div class="editor-label">
                @Html.LabelFor(model => model.CareerCode)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.CareerCode)
                @Html.ValidationMessageFor(model => model.CareerCode)
            </div>

            <div class="editor-label">
                @Html.LabelFor(model => model.CourseCode)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.CourseCode)
                @Html.ValidationMessageFor(model => model.CourseCode)
            </div>

            <div class="editor-label">
                @Html.LabelFor(model => model.YearInTheCareer)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.YearInTheCareer)
                @Html.ValidationMessageFor(model => model.YearInTheCareer)
            </div>
```

```
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.CareerCourseRelationship>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CareerCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.CourseCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.YearInTheCareer)
        </th>

        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CourseCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.YearInTheCareer)
        </td>
```

```
            <td>
                @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
                @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
                @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
            </td>
        </tr>
    }

    </table>
```

Course

Create.cshtml

```
    @model TesinaMobileCloud.Data.Model.Course

    @{
        ViewBag.Title = "Create";
        Layout = "~/Views/Shared/_Layout.cshtml";
    }

    <h2>Create</h2>

    @using (Html.BeginForm()) {
        @Html.ValidationSummary(true)

        <fieldset>
            <legend>Course</legend>

            <div class="editor-label">
                @Html.LabelFor(model => model.CourseCode)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.CourseCode)
                @Html.ValidationMessageFor(model => model.CourseCode)
            </div>

            <div class="editor-label">
                @Html.LabelFor(model => model.Title)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.Title)
                @Html.ValidationMessageFor(model => model.Title)
            </div>

            <div class="editor-label">
                @Html.LabelFor(model => model.Professor)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.Professor)
                @Html.ValidationMessageFor(model => model.Professor)
            </div>
            <p>
                <input type="submit" value="Create" />
            </p>
```

```
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<fieldset>
    <legend>Course</legend>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.CourseCode)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Title)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Professor)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Delete" /> |
        @Html.ActionLink("Back to List", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<fieldset>
    <legend>Course</legend>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.CourseCode)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.DisplayNameFor(model => model.Title)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
    <div class="display-label">
        @Html.DisplayNameFor(model => model.Professor)
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.CourseCode }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Course</legend>
```

```
            <div class="editor-label">
                @Html.LabelFor(model => model.CourseCode)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.CourseCode)
                @Html.ValidationMessageFor(model => model.CourseCode)
            </div>

            <div class="editor-label">
                @Html.LabelFor(model => model.Title)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.Title)
                @Html.ValidationMessageFor(model => model.Title)
            </div>

            <div class="editor-label">
                @Html.LabelFor(model => model.Professor)
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.Professor)
                @Html.ValidationMessageFor(model => model.Professor)
            </div>
            <p>
                <input type="submit" value="Save" />
            </p>
        </fieldset>
    }

    <div>
        @Html.ActionLink("Back to List", "Index")
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }
```

Index.cshtml

```
    @model IEnumerable<TesinaMobileCloud.Data.Model.Course>

    @{
        ViewBag.Title = "Index";
        Layout = "~/Views/Shared/_Layout.cshtml";
    }

    <h2>Index</h2>

    <p>
        @Html.ActionLink("Create New", "Create")
    </p>
    <table>
        <tr>
```

```
            <th>
                @Html.DisplayNameFor(model => model.CourseCode)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Professor)
            </th>
            <th></th>
        </tr>

    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.CourseCode)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Professor)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id=item.CourseCode }) |
                @Html.ActionLink("Details", "Details", new { id=item.CourseCode }) |
                @Html.ActionLink("Delete", "Delete", new { id=item.CourseCode, row = item.Title })
            </td>
        </tr>
    }

    </table>
```

Home

Index.cshtml

```
    @{
        ViewBag.Title = "Sitio de Administración - Tesina Carlos Rodrigo";
    }
    @section featured {
        <section class="featured">
            <div class="content-wrapper">
                <hgroup class="title">
                    <h1>@ViewBag.Title.</h1>
                    <h2>@ViewBag.Message</h2>
                </hgroup>
                <p>
                    Este sitio provee las funciones de administración y carga de datos necesarios
    para el desarrollo de la tesina.
                    La necesidad surge de simular los sistemas de asistencias y administración de
    los alumnos, para solo dedicarse a
                    la implementación del cliente Mobile.
                </p>
            </div>
```

```
    </section>
}
```

Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<hgroup class="title">
    <h1 class="error">Error.</h1>
    <h2 class="error">An error occurred while processing your request.</h2>
</hgroup>
```

Shared

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title - My ASP.NET MVC Application</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width" />
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
                </div>
                <div class="float-right">
                    <section id="login">
                        @Html.Partial("_LoginPartial")
                    </section>
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("Cátedras", "Index", "Course")</li>
                            <li>@Html.ActionLink("Carreras", "Index", "Career")</li>
                            <li>@Html.ActionLink("Cátedras-Carreras", "Index",
"CareerCourseRelationship")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
```

```
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
            </div>
        </footer>

        @Scripts.Render("~/bundles/jquery")
        @RenderSection("scripts", required: false)
    </body>
</html>
```

TesinaMobileCloud.Phone.Client

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone" xmlns:sys="clr-
namespace:System;assembly=mscorlib"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.ViewModel" mc:Ignorable="d">
  <!--Application Resources-->
  <Application.Resources>
    <ResourceDictionary>
      <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
      <ResourceDictionary.MergedDictionaries></ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
  <Application.ApplicationLifetimeObjects>
    <!--Required object that handles lifetime events for the application-->
    <shell:PhoneApplicationService Launching="Application_Launching"
                                   Closing="Application_Closing"
                                   Activated="Application_Activated"
                                   Deactivated="Application_Deactivated" />
  </Application.ApplicationLifetimeObjects>
</Application>
```

App.xaml.cs

```
using System.Windows;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using TesinaMobileCloud.Phone.Client.ViewModel;
```

```
namespace TesinaMobileCloud.Phone.Client
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        protected ViewModelLocator Locator
        {
            get { return (ViewModelLocator)Resources["Locator"]; }
        }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
                // application's PhoneApplicationService object to Disabled.
                // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
                // and consume battery power when the user is not using the phone.
                PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
            }
```

```
        }

        // Code to execute when the application is launching (eg, from Start)
        // This code will not execute when the application is reactivated
        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
            Locator.ScheduleActions.RemovePeriodicTask();
        }

        // Code to execute when the application is activated (brought to foreground)
        // This code will not execute when the application is first launched
        private void Application_Activated(object sender, ActivatedEventArgs e)
        {

        }

        // Code to execute when the application is deactivated (sent to background)
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEventArgs e)
        {
            // Ensure that required application state is persisted here.
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
        {
            Locator.ScheduleActions.AddPeriodicTask();
        }

        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // A navigation has failed; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization
```

```
        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

Views

MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="TesinaMobileCloud.Phone.Client.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:View="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="800"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
```

```
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"
    Orientation="Portrait"
    shell:SystemTray.IsVisible="False">
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <controls:Panorama Title="Sia">
            <controls:PanoramaItem Header="Cátedras">
                <View:CoursesView/>
            </controls:PanoramaItem>
        </controls:Panorama>
    </Grid>
</phone:PhoneApplicationPage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

CoursesView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    x:Class="TesinaMobileCloud.Phone.Client.View.CoursesView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="498" d:DesignWidth="420"
```

```
    DataContext="{Binding Courses, Source={StaticResource Locator}}">
    <Grid x:Name="Courses" Background="{StaticResource PhoneChromeBrush}">
        <toolkit:LongListSelector x:Name="Selector" ItemsSource="{Binding Courses}"
Background="Black" IsFlatList="True">
            <toolkit:LongListSelector.GroupItemTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding GroupName}" FontSize="32" Foreground="Green"
/>
                </DataTemplate>
            </toolkit:LongListSelector.GroupItemTemplate>
            <toolkit:LongListSelector.ItemTemplate>
                <DataTemplate>
                        <local:CourseView Height="85" Width="432"/>
                </DataTemplate>
            </toolkit:LongListSelector.ItemTemplate>
        </toolkit:LongListSelector>
    </Grid>
</UserControl>
```

CoursesView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CoursesView : UserControl
    {
        public CoursesView()
        {
            InitializeComponent();
        }
    }
}
```

CourseView.xaml

```
<UserControl x:Class="TesinaMobileCloud.Phone.Client.View.CourseView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
```

```
    d:DesignHeight="75" d:DesignWidth="432">

    <Grid x:Name="LayoutRoot" Margin="0,0,0,10">
        <Grid Height="75" VerticalAlignment="Top">
            <TextBlock x:Name="Title" HorizontalAlignment="Left" Margin="25,0,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Height="43" FontSize="32" Text="{Binding
Title}"/>
            <TextBlock x:Name="Professor" HorizontalAlignment="Left"
Margin="25,48,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Text="{Binding
Professor}"/>
            <Border x:Name="Attendance" Background="{Binding AttendanceState}"
HorizontalAlignment="Left" Height="75" VerticalAlignment="Top" Width="20"/>
        </Grid>
    </Grid>
</UserControl>
```

CourseView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CourseView : UserControl
    {
        public CourseView()
        {
            InitializeComponent();
        }
    }
}
```

ViewModels

CoursesViewModel.cs

```
using GalaSoft.MvvmLight;
using System.Collections.ObjectModel;
using System.Linq;
using TesinaMobileCloud.Phone.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CoursesViewModel : ViewModelBase
    {
        public CoursesViewModel(ICourseRepository courseRepository)
        {
            Courses = new ObservableCollection<CourseViewModel>();
```

```
            var courses = courseRepository.GetCourses();
            if (courses != null)
                courses.ToList().ForEach(c => Courses.Add(new CourseViewModel
                                                {
                                                    Title = c.Title,
                                                    Code = c.Code,
                                                    Professor = c.Professor,
                                                    Attendance = c.Attendance,
                                                    YearOfCareer = c.YearOfCareer
                                                }));
        }

        public ObservableCollection<CourseViewModel> Courses { get; private set; }
    }
}
```

CourseViewModel.cs

```
using GalaSoft.MvvmLight;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Media;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseViewModel : ViewModelBase
    {
        private int _code;
        private string _title;
        private int _yearOfCareer;
        private string _professor;
        private int _attendance;

        public int Code
        {
            get { return _code; }
            set
            {
                _code = value;
                RaisePropertyChanged("Code");
            }
        }
        public string Title
        {
            get { return _title; }
            set
            {
                _title = value;
                RaisePropertyChanged("Title");
            }
        }
        public int YearOfCareer
```

```
        {
            get { return _yearOfCareer; }
            set
            {
                _yearOfCareer = value;
                RaisePropertyChanged("YearOfCareer");
            }
        }
        public string Professor
        {
            get { return _professor; }
            set
            {
                _professor = value;
                RaisePropertyChanged("Professor");
            }
        }
        public int Attendance
        {
            get { return _attendance; }
            set
            {
                _attendance = value;
                RaisePropertyChanged("Attendance");
            }
        }
        public SolidColorBrush AttendanceState
        {
            get
            {
                switch (Attendance)
                {
                    case 1:
                        return new SolidColorBrush(Colors.Red);
                    case 2:
                        return new SolidColorBrush(Colors.Yellow);
                    case 3:
                        return new SolidColorBrush(Colors.Green);
                    default:
                        return new SolidColorBrush(Colors.White);
                }
            }
        }
    }
}
```

MainViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class MainViewModel : ViewModelBase
    {
```

```
        public MainViewModel()
        {
        }
    }
}
```

ViewModelLocator.cs

```
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;
using TesinaMobileCloud.Phone.Repositories;
using TesinaMobileCloud.Phone.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class ViewModelLocator
    {
        public ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            if (ViewModelBase.IsInDesignModeStatic)
            {
                // Create design time view services and models
                SimpleIoc.Default.Register<ICourseRepository, MockCourseRepository>();
                SimpleIoc.Default.Register<IScheduleActionClient, ScheduleActionClient>();
                SimpleIoc.Default.Register<IScheduleActionService,
ScheduleActionServiceAdapter>();
            }
            else
            {
                // Create run time view services and models
                SimpleIoc.Default.Register<ICourseRepository, CourseRepository>();
                SimpleIoc.Default.Register<IScheduleActionClient, ScheduleActionClient>();
                SimpleIoc.Default.Register<IScheduleActionService,
ScheduleActionServiceAdapter>();
            }

            SimpleIoc.Default.Register<MainViewModel>();
            SimpleIoc.Default.Register<CoursesViewModel>();
            SimpleIoc.Default.Register<CourseViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }

        public CoursesViewModel Courses
```

```
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CoursesViewModel>();
            }
        }

        public CourseViewModel Course
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CourseViewModel>();
            }

        }

        public IScheduleActionClient ScheduleActions
        {
            get
            {
                return ServiceLocator.Current.GetInstance<IScheduleActionClient>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

TesinaMobileCloud.Phone.Client.Test

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.Test.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test.ViewModel" mc:Ignorable="d">
  <!--Application Resources-->
  <Application.Resources>
    <ResourceDictionary>
      <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
      <ResourceDictionary.MergedDictionaries></ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
  <Application.ApplicationLifetimeObjects>
    <!--Required object that handles lifetime events for the application-->
```

```
    <shell:PhoneApplicationService Launching="Application_Launching"
Closing="Application_Closing" Activated="Application_Activated"
Deactivated="Application_Deactivated" />
  </Application.ApplicationLifetimeObjects>
</Application>
```

App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;
```

```
            // Enable non-production analysis visualization mode,
            // which shows areas of a page that are handed off to GPU with a colored
overlay.
            //Application.Current.Host.Settings.EnableCacheVisualization = true;

            // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
            // application's PhoneApplicationService object to Disabled.
            // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
            // and consume battery power when the user is not using the phone.
            PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
        }

    }

    // Code to execute when the application is launching (eg, from Start)
    // This code will not execute when the application is reactivated
    private void Application_Launching(object sender, LaunchingEventArgs e)
    {
    }

    // Code to execute when the application is activated (brought to foreground)
    // This code will not execute when the application is first launched
    private void Application_Activated(object sender, ActivatedEventArgs e)
    {
    }

    // Code to execute when the application is deactivated (sent to background)
    // This code will not execute when the application is closing
    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
    {
    }

    // Code to execute when the application is closing (eg, user hit Back)
    // This code will not execute when the application is deactivated
    private void Application_Closing(object sender, ClosingEventArgs e)
    {
    }

    // Code to execute if a navigation fails
    private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
    {
        if (System.Diagnostics.Debugger.IsAttached)
        {
            // A navigation has failed; break into the debugger
            System.Diagnostics.Debugger.Break();
        }
    }

    // Code to execute on Unhandled Exceptions
```

```
        private void Application_UnhandledException(object sender,
    ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
    e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

Views

MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="TesinaMobileCloud.Phone.Client.Test.MainPage"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Testing;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
```

```
        public MainPage()
        {
            InitializeComponent();
            this.Content = UnitTestSystem.CreateTestPage();
        }
    }
}
```

UnitTest

CoursesViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.ObjectModel;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Repositories;
using TesinaMobileCloud.Phone.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CoursesViewModelFixture
    {
        readonly ICourseRepository _coursesRepository = new MockCourseRepository();

        [TestMethod]
        public void CoursesViewModel()
        {
            var sut = new CoursesViewModel(_coursesRepository);

            Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
            Assert.IsInstanceOfType(sut.Courses,
typeof(ObservableCollection<CourseViewModel>));
        }

        [TestMethod]
        public void CoursesViewModelFillCoursesList()
        {
            var sut = new CoursesViewModel(_coursesRepository);

            Assert.IsNotNull(sut.Courses);
            Assert.AreEqual(1, sut.Courses.Count);
        }
    }
}
```

CourseViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Windows.Media;
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
   [TestClass]
   public class CourseViewModelFixture
   {
      [TestMethod]
      public void CourseViewModelWithParams()
      {
         var sut = new CourseViewModel
         {
            Code = 124,
            Title = "Title",
            Professor = "Prof",
            YearOfCareer = 1,
            Attendance = 1
         };

         Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
         Assert.IsNotNull(sut);
         Assert.AreEqual(124, sut.Code);
         Assert.AreEqual("Title", sut.Title);
         Assert.AreEqual(1, sut.YearOfCareer);
         Assert.AreEqual("Prof", sut.Professor);
         Assert.AreEqual(1, sut.Attendance);
         Assert.IsInstanceOfType(sut.AttendanceState, typeof(SolidColorBrush));
      }

      [TestMethod]
      public void CourseViewModelWithOutParams()
      {
         var sut = new CourseViewModel();
         Assert.IsNotNull(sut);
      }
   }
}
```

ViewModels

MainViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
   /// <summary>
   /// This class contains properties that the main View can data bind to.
   /// <para>
   /// Use the <strong>mvvminpc</strong> snippet to add bindable properties to this
ViewModel.
   /// </para>
   /// <para>
   /// You can also use Blend to data bind with the tool's support.
```

```
        /// </para>
        /// <para>
        /// See http://www.galasoft.ch/mvvm
        /// </para>
        /// </summary>
        public class MainViewModel : ViewModelBase
        {
            /// <summary>
            /// Initializes a new instance of the MainViewModel class.
            /// </summary>
            public MainViewModel()
            {
                ////if (IsInDesignMode)
                ////{
                ////    // Code runs in Blend --> create design time data.
                ////}
                ////else
                ////{
                ////    // Code runs "for real"
                ////}
            }
        }
    }
```

ViewModelLocator.cs

```
    using GalaSoft.MvvmLight;
    using GalaSoft.MvvmLight.Ioc;
    using Microsoft.Practices.ServiceLocation;

    namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
    {
        /// <summary>
        /// This class contains static references to all the view models in the
        /// application and provides an entry point for the bindings.
        /// </summary>
        public class ViewModelLocator
        {
            /// <summary>
            /// Initializes a new instance of the ViewModelLocator class.
            /// </summary>
            public ViewModelLocator()
            {
                ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

                ////if (ViewModelBase.IsInDesignModeStatic)
                ////{
                ////    // Create design time view services and models
                ////    SimpleIoc.Default.Register<IDataService, DesignDataService>();
                ////}
                ////else
                ////{
                ////    // Create run time view services and models
                ////    SimpleIoc.Default.Register<IDataService, DataService>();
                ////}
```

```
            SimpleIoc.Default.Register<MainViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

TesinaMobileCloud.Phone.Data

Course.cs

```
namespace TesinaMobileCloud.Phone.Data
{
    public class Course
    {
        public string Title { get;  set; }

        public int Code { get;  set; }

        public string Professor { get; set; }

        public int Attendance { get; set; }

        public int YearOfCareer { get; set; }
    }
}
```

TesinaMobileCloud.Phone.Agents

ScheduledAgent.cs

```
using System;
using System.Windows;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Repositories;
using TesinaMobileCloud.Phone.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;
using Microsoft.Phone.Shell;
using System.Linq;
```

```
namespace TesinaMobileCloud.Phone.Agents
{
    public class ScheduledAgent : ScheduledTaskAgent
    {
        private static volatile bool _classInitialized;
        private readonly ICourseRepository _courseRepository;
        private readonly ICloudService _cloudService;

        /// <remarks>
        /// ScheduledAgent constructor, initializes the UnhandledException handler
        /// </remarks>
        public ScheduledAgent()
        {
            if (!_classInitialized)
            {
                _classInitialized = true;
                // Subscribe to the managed exception handler
                Deployment.Current.Dispatcher.BeginInvoke(delegate
                {
                    Application.Current.UnhandledException +=
ScheduledAgent_UnhandledException;
                });
            }

            _courseRepository = new CourseRepository();
            _cloudService = new CloudService();
        }

        /// Code to execute on Unhandled Exceptions
        private void ScheduledAgent_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        /// <summary>
        /// Agent that runs a scheduled task
        /// </summary>
        /// <param name="task">
        /// The invoked task
        /// </param>
        /// <remarks>
        /// This method is called when a periodic or resource intensive task is invoked
        /// </remarks>
        protected override void OnInvoke(ScheduledTask task)
        {
            if (task is PeriodicTask)
            {
                PerformSyncronizationTask();
```

```
            }
        }
        private void PerformSyncronizationTask()
        {
            var syncService = new SyncronizationService(_cloudService, _courseRepository);

            syncService.SyncCourses()
                    .ObserveOnDispatcher()
                    .Subscribe(SyncCompleted, SyncFailed);
        }

        private void SyncCompleted(TaskSummary result)
        {
            DisplayShellNotification(new StandardTileData
                                {
                                    Title = "SIA",
                                    BackContent = "Información Actualizada"
                                });
            NotifyComplete();
        }

        private void SyncFailed(Exception exception)
        {
            DisplayShellNotification(new StandardTileData
                                {
                                    Title = "SIA",
                                    BackContent = "Error al sincronizar datos"
                                });
            Abort();
        }

        private static void DisplayShellNotification(ShellTileData tileData)
        {
            var tile = ShellTile.ActiveTiles.First();
            tile.Update(tileData);
        }
    }
}
```

TesinaMobileCloud.Phone.Repositories

CourseRepository.cs

```
using System.Collections.Generic;
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Phone.Data;
using TesinaMobileCloud.Phone.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Repositories
{
    public class CourseRepository : ICourseRepository
    {
        private readonly IsolatedStorageSettings _isolatedStorage;
```

```
        public CourseRepository()
        {
            _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
        }

        public IEnumerable<Course> GetCourses()
        {
            IEnumerable<Course> courses;
            _isolatedStorage.TryGetValue("Courses", out courses);
            return courses;
        }

        public void AddCourses(IEnumerable<Course> courses)
        {
            IEnumerable<Course> coursesCollection;
            if (_isolatedStorage.TryGetValue("Courses", out coursesCollection))
            {
                _isolatedStorage.Remove("Courses");
            }
            _isolatedStorage.Add("Courses", courses);
            _isolatedStorage.Save();
        }
    }
}
```

MockCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Phone.Data;
using TesinaMobileCloud.Phone.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Repositories
{
    public class MockCourseRepository : ICourseRepository
    {
        private static List<Course> _courses;

        public MockCourseRepository()
        {
            _courses = new List<Course>();
        }

        public IEnumerable<Course> GetCourses()
        {
            return _courses;
        }

        public void AddCourses(IEnumerable<Course> courses)
        {
            foreach (var course in courses)
                _courses.Add(course);
        }
    }
}
```

ICourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Phone.Data;

namespace TesinaMobileCloud.Phone.Repositories.Interfaces
{
    public interface ICourseRepository
    {
        IEnumerable<Course> GetCourses();
        void AddCourses(IEnumerable<Course> courses);
    }
}
```

TesinaMobileCloud.Phone.Repositories.Test

ICourseRepositoryFixture.cs

```
using System.Collections.Generic;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Data;

namespace TesinaMobileCloud.Phone.Repositories.Test
{
    [TestClass]
    public class CourseRepositoryFixture
    {
        [TestMethod]
        public void GetCourses()
        {
            var sut = new MockCourseRepository();

            var result = sut.GetCourses();

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(IEnumerable<Data.Course>));
        }

        [TestMethod]
        public void AddCourses()
        {
            var courses = new List<Course>
                        {
                            new Course
                            {
                                Attendance = 1, Title = "Programacion I", Professor = "Prof.
Diana Ciccinelli"
                            },
                            new Course
                            {
                                Attendance = 2, Title = "Programacion II", Professor = "Prof.
Martin Cernadas"
                            }
                        };
```

```
            var sut = new MockCourseRepository();

            sut.AddCourses(courses);

            var result = sut.GetCourses();
            Assert.IsNotNull(result);
            Assert.AreEqual(2, result.Count());
        }
    }
}
```

TesinaMobileCloud.Phone.Services

CloudService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Runtime.Serialization.Json;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class CloudService : ICloudService
    {
        public IObservable<IEnumerable<Course>> GetAllCourses(Uri baseUri)
        {
            var relativeUri = String.Format("RetriveCoursesByCareer/{0}", "502");//TODO: Alto
HardCoding, que lo tome del IsolatedStorage

            var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
            request.Method = "GET";
            request.Accept = "application/json";

            return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
                .Select(
                    response =>
                        {
                            using (var responseStream = response.GetResponseStream())
                            {
                                var serializer = new
DataContractJsonSerializer(typeof(IEnumerable<CourseDTO>));
                                return serializer.ReadObject(responseStream) as
IEnumerable<CourseDTO>;
                            }
                        }
                ).Select(ToCourseCollection);
        }
```

```
        private static IEnumerable<Course> ToCourseCollection(IEnumerable<CourseDTO>
    course)
        {
            return course.Select(courseDto => new Course
                                               {
                                                   Attendance = courseDto.Attendance,
                                                   Title = courseDto.Title,
                                                   Professor = courseDto.Professor,
                                                   Code = courseDto.CourseCode,
                                                   YearOfCareer = courseDto.YearOfCareer
                                               }).ToList();
        }
    }
}
```

ScheduleActionClient.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionClient : IScheduleActionClient
    {
        private const string PeriodicTaskName = "SyncCourses";
        private readonly IScheduleActionService _scheduleActionService;

        public ScheduleActionClient(IScheduleActionService scheduleActionService)
        {
            _scheduleActionService = scheduleActionService;
        }

        public void AddPeriodicTask()
        {
            var periodicTask = new PeriodicTask(PeriodicTaskName)
                               {
                                   Description = "Syncronization Courses information Task"
                               };
            if (_scheduleActionService.Find(PeriodicTaskName) != null)
                _scheduleActionService.Remove(PeriodicTaskName);

            _scheduleActionService.Add(periodicTask);

#if DEBUG
            _scheduleActionService.LaunchForTest(PeriodicTaskName,
    TimeSpan.FromMinutes(1));
#endif
        }

        public void RemovePeriodicTask()
        {
            if (_scheduleActionService.Find(PeriodicTaskName) != null)
                _scheduleActionService.Remove(PeriodicTaskName);
        }
```

```
    }
}
```

ScheduleActionServiceAdapter.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionServiceAdapter : IScheduleActionService
    {
        public ScheduledAction Find(string taskName)
        {
            return ScheduledActionService.Find(taskName);
        }

        public void Add(ScheduledAction action)
        {
            ScheduledActionService.Add(action);
        }

        public void Remove(string taskName)
        {
            ScheduledActionService.Remove(taskName);
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            ScheduledActionService.LaunchForTest(actionName, delay);
        }
    }
}
```

SyncronizationService.cs

```
using System;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Phone.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services
{
    public class SyncronizationService : ISyncronizationService
    {
        private readonly ICloudService _cloudService;
        private readonly ICourseRepository _courseRepository;

        public SyncronizationService(ICloudService cloudService, ICourseRepository
courseRepository)
        {
            _cloudService = cloudService;
            _courseRepository = courseRepository;
        }
```

```
        public IObservable<TaskSummary> SyncCourses()
        {
//#if ONLY_PHONE
            var baseUri = new Uri("http://168.62.181.217:8081/Service1.svc/");
//#else
 //           var baseUri = new Uri("http://tesinamobilecloud.cloudapp.net");
//#endif
            var resuls = _cloudService.GetAllCourses(baseUri)
                .Select(courses =>
                        {
                            _courseRepository.AddCourses(courses);
                            return new TaskSummary
                                    {
                                        Result = TaskResult.Success
                                    };
                        })
                .Catch((Exception exception) =>
                        {
                            throw exception;
                        });

            return resuls;
        }
    }
}
```

ICloudService.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using TesinaMobileCloud.Phone.Data;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ICloudService
    {
        IObservable<IEnumerable<Course>> GetAllCourses(Uri baseUri);
    }
}
```

IScheduleActionClient.cs

```
namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionClient
    {
        void AddPeriodicTask();
        void RemovePeriodicTask();
    }
}
```

IScheduleActionService.cs

```
using System;
using Microsoft.Phone.Scheduler;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionService
    {
        ScheduledAction Find(string taskName);
        void Add(ScheduledAction action);
        void Remove(string taskName);
        void LaunchForTest(string actionName, TimeSpan delay);
    }
}
```

ISyncronizationService.cs

```
using System;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ISyncronizationService
    {
        IObservable<TaskSummary> SyncCourses();
    }
}
```

MockCloudService.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Phone.Data;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockCloudService : ICloudService
    {
        public IObservable<IEnumerable<Course>> GetAllCourses(Uri baseUri)
        {
            return null;
        }
    }
}
```

MockScheduleActionService.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockScheduleActionService : IScheduleActionService
    {
```

```
        public ScheduledAction Find(string taskName)
        {
            throw new NotImplementedException();
        }

        public void Add(ScheduledAction action)
        {
            throw new NotImplementedException();
        }

        public void Remove(string taskName)
        {
            throw new NotImplementedException();
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            throw new NotImplementedException();
        }
    }
}
```

MockSyncronizationService.cs

```
using System;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockSyncronizationService : ISyncronizationService
    {
        public IObservable<TaskSummary> SyncCourses()
        {
            return Observable.Return(new TaskSummary
                                     {
                                         Result = TaskResult.Success
                                     });
        }
    }
}
```

TaskResult.cs

```
namespace TesinaMobileCloud.Phone.Services.Support
{
    public enum TaskResult
    {
        Success,
        Failed
    }
}
```

TaskSummary.cs

```
namespace TesinaMobileCloud.Phone.Services.Support
{
    public class TaskSummary
    {
        public TaskResult Result { get; set; }
    }
}
```

TesinaMobileCloud.Phone.Services.Test

CloudServiceFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class CloudServiceFixture
    {
        [TestMethod]
        public void GetAllCourses()
        {
            var sut = new MockCloudService();
            var result = sut.GetAllCourses(new Uri("http://127.0.0.1"));

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(IEnumerable<Data.Course>));
        }
    }
}
```

ScheduleActionsServiceFixture.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class ScheduleActionsServiceFixture
    {
        [TestMethod]
        public void RemoveAgent()
        {
            var sut = new ScheduleActionClient(new MockScheduleActionService());

            sut.RemovePeriodicTask();

        }
```

```
        [TestMethod]
        public void AddAgent()
        {
            var sut = new ScheduleActionClient(new MockScheduleActionService());

            sut.AddPeriodicTask();

        }
    }
}
```

SyncronizationServiceFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class SyncronizationServiceFixture
    {
        [TestMethod]
        public void SyncCourses()
        {
            var sut = new MockSyncronizationService();

            var result = sut.SyncCourses();

            Assert.IsNotNull(result);
            //Assert.IsInstanceOfType(result, typeof(IObservable<TaskSummary>));
        }
    }
}
```

TesinaMobileCloud.Shared

ServiceModule.cs

```
using Microsoft.WindowsAzure;
using Ninject.Modules;
using TesinaMobileCloud.Data;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Shared
{
    public class ServiceModule : NinjectModule
    {
        public override void Load()
```

```
        {
            Bind<CloudStorageAccount>().ToMethod(context =>
CloudStorageConfiguration.GetCloudAccount("DataConnectionString"));

            Bind<ITable<Course>>().To<AzureTable<Course>>();
            Bind<ITable<Career>>().To<AzureTable<Career>>();

Bind<ITable<CareerCourseRelationship>>().To<AzureTable<CareerCourseRelationship>>(
);

            Bind<ICourseRepository>().To<CourseRepository>();

Bind<ICourseCareerRelationshipRepository>().To<CareerCourseRelationshipRepository>()
;
        }
    }
}
```

AttendanceState.cs

```
namespace TesinaMobileCloud.Shared
{
    public enum AttendanceState
    {
        Danger = 1,
        Precaution = 2,
        Good = 3
    }
}
```

3. Anexo 3 – Iteración 3

ServiceRoleTest

IServiceRoleFixture.cs

```
using System;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using Course = TesinaMobileCloud.Data.DTO.Course;
using CourseAzure = TesinaMobileCloud.Data.Model.Course;
using Student = TesinaMobileCloud.Data.Model.Student;
using TesinaMobileCloud.Data.DTO;

namespace ServiceRoleTest
{
   [TestClass]
   public class ServiceRoleFixture
   {
      const string CareerCode = "502";
      const string StudentCode = "10004";
      const int CourseCode = 120;

      private Mock<ICourseRepository> _repositoryCourse;
      private Mock<ICareerRepository> _repositoryCareer;
      private Mock<ICourseCareerRelationshipRepository> _repositoryRelationship;
      private Mock<IStudentRepository> _student;
      private Mock<IStudentCourseRepository> _studentCourse;

      [TestInitialize]
      public void Setup()
      {
         _repositoryCourse = new Mock<ICourseRepository>();
         _repositoryCareer = new Mock<ICareerRepository>();
         _repositoryRelationship = new Mock<ICourseCareerRelationshipRepository>();
         _repositoryCourse.Setup(m => m.GetCoursesByCareer(CareerCode)).Returns(new
List<TesinaMobileCloud.Data.Model.Course>
                                                {
                                                   new
TesinaMobileCloud.Data.Model.Course(CourseCode,"Programación I")
                                                   {
                                                      Professor = "Mg. Angeleri, Paula",
                                                   }
                                                });
         _repositoryRelationship.Setup(m =>
m.GetCareerYearOfCourseByCareer(int.Parse(CareerCode),
                                                   CourseCode))
                                .Returns(1);
         _student = new Mock<IStudentRepository>();
         _studentCourse = new Mock<IStudentCourseRepository>();
```

```
        }


        [TestMethod]
        public void RetriveStudentByStudentCodeAndCareer()
        {
            _student.Setup(m =>
m.GetStudentByCareerCodeAndStudentCode(It.IsAny<string>(), It.IsAny<string>()))
                    .Returns(new Student
                    {
                        CareerCode = 502,
                        StudentCode = 10004,
                        FirstName = "Carlos",
                        LastName = "Rodrigo"
                    });
            _studentCourse.Setup(m => m.RetriveCoursesOfStudent(It.IsAny<string>()))
                        .Returns(new List<StudentCourseRelationship>
                        {
                            new StudentCourseRelationship
                            {
                                CourseCode = 1,
                            }
                        });
            _repositoryCourse.Setup(m => m.GetSingleCourse(It.IsAny<int>()))
                        .Returns(new CourseAzure
                        {
                            Code = 1,
                            Title = "Ing. del Software",
                            Professor = "Mg. Paula Angeleri",
                            TotalHoursOfClasses = 60,
                            Description = "Lalo",
                            Scheduled = "Miercoles 17Hs",
                            FinalExamDate = new DateTime(2013, 12, 22),
                            PartialExameDate = new DateTime(2013, 10, 10),
                            PracticWorkPresentationDate = new DateTime(2013, 12, 20),
                            RecuperationExameDate = new DateTime(2013, 11, 20)
                        });
            _repositoryCareer.Setup(m => m.GetCareer(It.IsAny<string>())).Returns(new
TesinaMobileCloud.Data.Model.Career());

            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object, _student.Object, _studentCourse.Object,
_repositoryCareer.Object);

            var result = sut.RetriveStudentByCodeAndCareer(StudentCode, CareerCode);

            Assert.IsNotNull(result);
            Assert.IsNotNull(result.Courses);
            Assert.AreEqual(1, result.Courses.Count());
            Assert.IsNotNull(result.Courses.First().PartialExamResult);
            Assert.IsNotNull(result.Courses.First().FinalExamResult);
            Assert.IsNotNull(result.Courses.First().RecuperationExamResult);
            Assert.IsNotNull(result.Courses.First().PracticWorkResult);
            Assert.IsNotNull(result.Courses.First().TotalHoursAssisted);
```

```
            Assert.IsNotNull(result.Courses.First().Course.TotalHoursOfClasses);
            Assert.IsNotNull(result.Courses.First().Course.Title);
            Assert.IsNotNull(result.Courses.First().Course.Professor);
            Assert.IsNotNull(result.Courses.First().Course.Code);
            Assert.IsNotNull(result.Courses.First().Course.Scheduled);
            Assert.IsNotNull(result.Courses.First().Course.Description);
            Assert.IsNotNull(result.Courses.First().Course.PartialExamDate);
            Assert.IsNotNull(result.Courses.First().Course.FinalExamDate);
            Assert.IsNotNull(result.Courses.First().Course.RecuperationExameDate);
            Assert.IsNotNull(result.Courses.First().Course.PracticWorkPresentationDate);
            Assert.IsNotNull(result.Career);
        }

        [TestMethod]
        public void RetriveStudentCourse()
        {
            var studentCode ="10004";
            var courseCode = "22";
            var careerCode = "502";
            _studentCourse.Setup(m => m.RetriveStudentCourseByCode(It.IsAny<string>(),
It.IsAny<string>()))
                        .Returns(
                              new StudentCourseRelationship
                                 {
                                    CourseCode = 1,
                                 }
                          );
            _repositoryCourse.Setup(m => m.GetSingleCourse(It.IsAny<int>()))
                         .Returns(new CourseAzure
                         {
                            Code = 1,
                            Title = "Ing. del Software",
                            Professor = "Mg. Paula Angeleri",
                            TotalHoursOfClasses = 60,
                            Description = "Lalo",
                            Scheduled = "Miercoles 17Hs",
                            FinalExamDate = new DateTime(2013, 12, 22),
                            PartialExameDate = new DateTime(2013, 10, 10),
                            PracticWorkPresentationDate = new DateTime(2013, 12, 20),
                            RecuperationExameDate = new DateTime(2013, 11, 20)
                         });

            var sut = new SiaService.SiaService(_repositoryCourse.Object,
                                        _repositoryRelationship.Object,
                                        _student.Object, _studentCourse.Object,
                                        _repositoryCareer.Object);

            var result = sut.RetriveStudentCourse(careerCode, studentCode, courseCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(StudentCourse));
        }
    }
}
```

SiaService

AzureLocalStorageTraceListener.cs

```
using System;
using System.Diagnostics;
using System.IO;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
   public class AzureLocalStorageTraceListener : XmlWriterTraceListener
   {
      public AzureLocalStorageTraceListener()
         : base(Path.Combine(AzureLocalStorageTraceListener.GetLogDirectory().Path,
"SiaService.svclog"))
      {
      }

      public static DirectoryConfiguration GetLogDirectory()
      {
         var directory = new DirectoryConfiguration
                                    {
                                       Container = "wad-tracefiles",
                                       DirectoryQuotaInMB = 10,
                                       Path =

RoleEnvironment.GetLocalResource("SiaService.svclog").RootPath
                                    };
         return directory;
      }
   }
}
```

Global.asax.cs

```
using System;
using Ninject;
using Ninject.Web.Common;
using TesinaMobileCloud.Shared;

namespace SiaService
{
   public class Global : NinjectHttpApplication
   {
      protected void Application_Start(object sender, EventArgs e)
      {
         Application_Start();
      }

      protected override IKernel CreateKernel()
      {
```

```
            return new StandardKernel(new ServiceModule());
        }
    }
}
```

ISiaService.cs

```
using System.ServiceModel;
using System.ServiceModel.Web;
using TesinaMobileCloud.Data.DTO;
using Student = TesinaMobileCloud.Data.DTO.Student;

namespace SiaService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "ISiaService" in both code and config file together.
    [ServiceContract]
    public interface ISiaService
    {

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RetriveStudentByCodeAndCareer/{studentCode}/{careerCode}")]
        Student RetriveStudentByCodeAndCareer(string studentCode, string careerCode);

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RetriveStudentCourse/{careerCode}/{studentCode}/{courseCode}")]
        StudentCourse RetriveStudentCourse(string careerCode, string studentCode, string
courseCode);
    }
}
```

Service1.svc.cs

```
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using CareerDto = TesinaMobileCloud.Data.DTO.Career;
using Course = TesinaMobileCloud.Data.DTO.Course;
using Student = TesinaMobileCloud.Data.DTO.Student;

namespace SiaService
{
    public class SiaService : ISiaService
    {
        private readonly ICareerRepository _careerRepository;
        private readonly ICourseRepository _courseRepository;
        private readonly ICourseCareerRelationshipRepository
_courseCareerRelationshipRepository;
        private readonly IStudentRepository _studentRepository;
        private readonly IStudentCourseRepository _studentCourseRelationshipRepository;
```

```
        public SiaService(ICourseRepository courseRepository,
ICourseCareerRelationshipRepository courseCareerRelationshipRepository,
IStudentRepository studentRepository, IStudentCourseRepository
studentCourseRelationshipRepository, ICareerRepository careerRepository)
        {
            _courseRepository = courseRepository;
            _courseCareerRelationshipRepository = courseCareerRelationshipRepository;
            _studentRepository = studentRepository;
            _studentCourseRelationshipRepository = studentCourseRelationshipRepository;
            _careerRepository = careerRepository;
        }

        public Student RetriveStudentByCodeAndCareer(string studentCode, string
careerCode)
        {
            var studentAzure =
_studentRepository.GetStudentByCareerCodeAndStudentCode(studentCode, careerCode);
            var studentToRetrive = new Student
                {
                    FirstName = studentAzure.FirstName,
                    LastName = studentAzure.LastName,
                    StudentCode = studentAzure.StudentCode,
                    CareerCode = studentAzure.CareerCode
                };
            var coursesOfStudent =
_studentCourseRelationshipRepository.RetriveCoursesOfStudent(studentCode);
            foreach (var studentCourseRelationship in coursesOfStudent)
            {
                var singleCourse =
_courseRepository.GetSingleCourse(studentCourseRelationship.CourseCode);
                studentToRetrive.Courses.Add(BuildStudentCourseDto(careerCode,
studentCourseRelationship, singleCourse));
            }

            var career = _careerRepository.GetCareer(careerCode);
            studentToRetrive.Career = new CareerDto
                {
                    Code = career.CareerCode.ToString(),
                    Title = career.Title
                };

            return studentToRetrive;
        }

        private StudentCourse BuildStudentCourseDto(string careerCode,
StudentCourseRelationship studentCourseRelationship,
TesinaMobileCloud.Data.Model.Course singleCourse)
        {
            return new StudentCourse
                {
                    FinalExamResult = studentCourseRelationship.FinalExamResult,
                    PartialExamResult = studentCourseRelationship.PartialExamResult,
                    PracticWorkResult = studentCourseRelationship.PracticWorkResult,
```

```
                RecuperationExamResult =
studentCourseRelationship.RecuperationExamResult,
                TotalHoursAssisted = studentCourseRelationship.TotalHoursAssisted,
                Course = new Course
                   {
                      Code = singleCourse.Code,
                      Professor = singleCourse.Professor,
                      Title = singleCourse.Title,
                      FinalExamDate = singleCourse.FinalExamDate,
                      PartialExamDate = singleCourse.PartialExameDate,
                      PracticWorkPresentationDate =
singleCourse.PracticWorkPresentationDate,
                      RecuperationExameDate = singleCourse.RecuperationExameDate,
                      Scheduled = singleCourse.Scheduled,
                      Description = singleCourse.Description,
                      YearOfCareer =
_courseCareerRelationshipRepository.GetCareerYearOfCourseByCareer(int.Parse(career
Code), singleCourse.Code),
                      TotalHoursOfClasses = singleCourse.TotalHoursOfClasses
                   }
             };
      }

      public StudentCourse RetriveStudentCourse(string careerCode, string studentCode,
string courseCode)
      {
         var studentCourse =
_studentCourseRelationshipRepository.RetriveStudentCourseByCode(studentCode,
courseCode);
         var course = _courseRepository.GetSingleCourse(studentCourse.CourseCode);

         return  BuildStudentCourseDto(careerCode, studentCourse, course);
      }
   }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
   public class WebRole : RoleEntryPoint
   {
      public override bool OnStart()
      {
         // To enable the AzureLocalStorageTraceListner, uncomment relevent section in the
web.config
```

```
            DiagnosticMonitorConfiguration diagnosticConfig =
DiagnosticMonitor.GetDefaultInitialConfiguration();
            diagnosticConfig.Directories.ScheduledTransferPeriod =
TimeSpan.FromMinutes(1);

diagnosticConfig.Directories.DataSources.Add(AzureLocalStorageTraceListener.GetLogDir
ectory());

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

TesinaMobileCloud.AdminSite

Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
```

```
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.AdminSite
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

BundleConfig.cs

```
using System.Web;
using System.Web.Optimization;

namespace TesinaMobileCloud.AdminSite
{
    public class BundleConfig
    {
        // For more information on Bundling, visit
http://go.microsoft.com/fwlink/?LinkId=254725
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                        "~/Scripts/jquery-ui-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then,
when you're
            // ready for production, use the build tool at http://modernizr.com to pick only the
tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

            bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
                        "~/Content/themes/base/jquery.ui.core.css",
                        "~/Content/themes/base/jquery.ui.resizable.css",
                        "~/Content/themes/base/jquery.ui.selectable.css",
                        "~/Content/themes/base/jquery.ui.accordion.css",
                        "~/Content/themes/base/jquery.ui.autocomplete.css",
```

```
                    "~/Content/themes/base/jquery.ui.button.css",
                    "~/Content/themes/base/jquery.ui.dialog.css",
                    "~/Content/themes/base/jquery.ui.slider.css",
                    "~/Content/themes/base/jquery.ui.tabs.css",
                    "~/Content/themes/base/jquery.ui.datepicker.css",
                    "~/Content/themes/base/jquery.ui.progressbar.css",
                    "~/Content/themes/base/jquery.ui.theme.css"));
        }
    }
}
```

FilterConfig.cs

```
using System.Web;
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

NinjectWebCommon.cs

```
using Ninject.Modules;
using TesinaMobileCloud.Shared;

[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NinjectWebCommon), "Start")]
[assembly:
WebActivator.ApplicationShutdownMethodAttribute(typeof(TesinaMobileCloud.AdminSite.A
pp_Start.NinjectWebCommon), "Stop")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;

    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
```

```
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var modules = new INinjectModule[] { new ServiceModule() };
            var kernel = new StandardKernel(modules);
            kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new Bootstrapper().Kernel);
            kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

            RegisterServices(kernel);
            return kernel;
        }

        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
}
```

NotificationManagementUi.cs

```
[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NotificationManagementUi), "PreStart")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System.Reflection;
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using TesinaMobileCloud.AdminSite.Areas.Notifications.Helpers;

    public static class NotificationManagementUi
```

```
{
    public const string TileImagesContainerName = "tileimagescontainer";

    public static void PreStart()
    {
        NotificationsManagementUiContext.Configure = c =>
            {
                // TODO: Replace with your own Windows Azure Storage account name and
key, or read it from a configuration file
                c.CloudStorageAccount = CloudStorageAccount.DevelopmentStorageAccount;

                // TODO: Replace with your prefered container name to store tile images
                c.TileImagesContainerName = TileImagesContainerName;

                return c;
            };


UploadTileImage("TesinaMobileCloud.AdminSite.Resources.WindowsAzureLogo.png");

UploadTileImage("TesinaMobileCloud.AdminSite.Resources.WindowsPhoneLogo.png");

UploadTileImage("TesinaMobileCloud.AdminSite.Resources.AzureBackground.png");

UploadTileImage("TesinaMobileCloud.AdminSite.Resources.DefaultBackground.png");
    }

    private static void UploadTileImage(string imageName)
    {
        var cloudBlobClient =
NotificationsManagementUiContext.Current.CloudStorageAccount.CreateCloudBlobClient()
;
        var tileImagesContainerName =
NotificationsManagementUiContext.Current.TileImagesContainerName;

        // Create the container (and make it public)
        var container = cloudBlobClient.GetContainerReference(tileImagesContainerName);
        container.CreateIfNotExist();
        container.SetPermissions(
            new BlobContainerPermissions
            {
                PublicAccess = BlobContainerPublicAccessType.Blob
            });

        // Upload the image from the assembly resources.
        var assembly = Assembly.GetExecutingAssembly();
        var imageStream = assembly.GetManifestResourceStream(imageName);
        var blob =
container.GetBlobReference(imageName.Replace("TesinaMobileCloud.AdminSite.Resourc
es.", string.Empty));
        blob.Properties.ContentType = "image/png";
        blob.UploadFromStream(imageStream);
    }
}
```

```
}
```

RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

WebApiConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace TesinaMobileCloud.AdminSite
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Controllers

CareerController.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerController : Controller
    {
        private readonly ITable<Career> _careers;

        public CareerController(ITable<Career> careers)
        {
            _careers = careers;
        }

//
        // GET: /Career/

        public ActionResult Index()
        {
            return View(_careers.GetByCriteria(career =>
!string.IsNullOrEmpty(career.PartitionKey)));
        }

        //
        // GET: /Career/Details/5

        public ActionResult Details(int id)
        {
            return View(_careers.GetSingleByCriteria(career => career.CareerCode == id));
        }

        //
        // GET: /Career/Create

        public ActionResult Create()
        {
            return View(new Career());
        }

        //
        // POST: /Career/Create

        [HttpPost]
        public ActionResult Create(Career career)
        {
            try
            {
                _careers.Add(career);
                return RedirectToAction("Index");
            }
            catch
```

```
            {
                return View();
            }
        }

        //
        // GET: /Career/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_careers.GetSingleByCriteria(career1 => career1.CareerCode == id));
        }

        //
        // POST: /Career/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var career = _careers.GetSingleByCriteria(career1 => career1.CareerCode ==
id);
                UpdateModel(career);
                _careers.Add(career);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Delete/5

        public ActionResult Delete(int id)
        {
            return View();
        }

        //
        // POST: /Career/Delete/5

        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                _careers.Delete(_careers.GetSingleByCriteria(c => c.CareerCode == id));

                return RedirectToAction("Index");
            }
```

```
            catch
            {
                return View();
            }
        }
    }
}
```

CareerCourseRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerCourseRelationshipController : Controller
    {
        private readonly ITable<CareerCourseRelationship> _careerCourseRelationship;

        public CareerCourseRelationshipController(ITable<CareerCourseRelationship>
careerCourseRelationship)
        {
            _careerCourseRelationship = careerCourseRelationship;
        }

//
        // GET: /CareerCourseRelationship/

        public ActionResult Index()
        {
            return View(_careerCourseRelationship.GetByCriteria(relationship =>
relationship.CareerCode != 0 ));
        }

        //
        // GET: /CareerCourseRelationship/Details/5

        public ActionResult Details(int course, int career)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CourseCode == course && relationship.CareerCode == career));
        }

        //
        // GET: /CareerCourseRelationship/Create

        public ActionResult Create()
        {
            return View(new CareerCourseRelationship());
```

```
        }

        //
        // POST: /CareerCourseRelationship/Create

        [HttpPost]
        public ActionResult Create(CareerCourseRelationship careerCourseRelationship)
        {
            try
            {
                _careerCourseRelationship.Add(careerCourseRelationship);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /CareerCourseRelationship/Edit/5

        public ActionResult Edit(int course, int career)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CourseCode == course && relationship.CareerCode == career));
        }

        //
        // POST: /CareerCourseRelationship/Edit/5

        [HttpPost]
        public ActionResult Edit(int careerCode, int courseCode, FormCollection collection)
        {
            try
            {
                var courseInCloud = _careerCourseRelationship.GetSingleByCriteria(s =>
s.CareerCode == careerCode && s.CourseCode == courseCode);
                UpdateModel(courseInCloud);
                _careerCourseRelationship.Add(courseInCloud);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /CareerCourseRelationship/Delete/5

        public ActionResult Delete(int careerCode, int courseCode)
        {
```

```
            return View(_careerCourseRelationship.GetSingleByCriteria(s => s.CareerCode ==
careerCode && s.CourseCode == courseCode));
        }

        //
        // POST: /CareerCourseRelationship/Delete/5

        [HttpPost]
        public ActionResult Delete(int careerCode, int courseCode, FormCollection collection)
        {
            try
            {

_careerCourseRelationship.Delete(_careerCourseRelationship.GetSingleByCriteria(c =>
c.CareerCode == careerCode && c.CourseCode == courseCode));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CourseController.cs

```
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CourseController : Controller
    {
        private readonly ITable<Course> _courses;

        public CourseController(ITable<Course> courses)
        {
            _courses = courses;
        }

        public ActionResult Index()
        {
            return View(_courses.GetByCriteria(course =>
!string.IsNullOrEmpty(course.PartitionKey)));
        }

        //
        // GET: /Course/Details/5

        public ActionResult Details(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.Code == id));
```

```
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new Course(0, string.Empty));
        }

        //
        // POST: /Course/Create

        [HttpPost]
        public ActionResult Create(Course course)
        {
            try
            {
                _courses.Add(course);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.Code == id));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _courses.GetSingleByCriteria(course => course.Code ==
id);
                UpdateModel(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
```

```
        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id)
        {
            try
            {
                _courses.Delete( _courses.GetSingleByCriteria(c => c.PartitionKey == id));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // POST: /Course/Delete/5


    }
}
```

HomeController.cs

```
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Sitio de administración y soporte, Tesina Carlos Rodrigo";

            return View();
        }
    }
}
```

StudentCodeRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Web;
using System.Web.Helpers;
using System.Web.Mvc;
using Newtonsoft.Json;
using TesinaMobileCloud.Data.Helpers;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Data.Table;
using TesinaMobileCloud.Notifications;
```

```
namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class StudentCodeRelationshipController : Controller
    {
        private readonly ITable<StudentCourseRelationship> _table;
        private readonly ITable<Course> _courses;
        private readonly ITable<Student> _students;
        private readonly IQueue<QueueNotificationMessage> _queueNotifications;

        public StudentCodeRelationshipController(ITable<StudentCourseRelationship> table,
ITable<Course> courses, IQueue<QueueNotificationMessage> queueNotifications,
ITable<Student> students)
        {
            _table = table;
            _courses = courses;
            _queueNotifications = queueNotifications;
            _students = students;
        }

        public ActionResult Index()
        {
            return View(_table.GetByCriteria(relationship => relationship.StudentCode != 0));
        }

        //
        // GET: /StudentCodeRelationship/Details/5

        public ActionResult Details(int id, int courseCode)
        {
            return View(_table.GetSingleByCriteria(s => s.StudentCode == id && s.CourseCode
== courseCode));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new StudentCourseRelationship());
        }

        //
        // POST: /Course/Create

        [HttpPost]
        public ActionResult Create(StudentCourseRelationship student)
        {
            try
            {
                _table.Add(student);
                return RedirectToAction("Index");
            }
            catch
```

```
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id, int courseCode)
        {
            return View(_table.GetSingleByCriteria(s => s.StudentCode == id && s.CourseCode
== courseCode));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, int courseCode, FormCollection collection)
        {
            try
            {
                var courseInCloud = _table.GetSingleByCriteria(s => s.StudentCode == id &&
s.CourseCode == courseCode);
                UpdateModel(courseInCloud);
                _table.Add(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
                _table.Delete(_table.GetSingleByCriteria(c => c.PartitionKey == id && c.RowKey
== row));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        public ActionResult ExamResultAsign(int id, int course)
        {
            try
```

```
        {
            ViewData["ExamTypes"] = new SelectList
                (
                  new List<SelectListItem>
                      {
                          new SelectListItem{ Text = "Parcial", Value="PartialExam"},
                          new SelectListItem{ Text = "Recuperatorio",
Value="RecuperationExam"},
                          new SelectListItem{ Text = "Final", Value="FinalExam"},
                          new SelectListItem{ Text = "Presentación de Trabajos Practicos",
Value="PracticalWorkExam"},
                      }
                );

            return View(_table.GetSingleByCriteria(s => s.StudentCode == id &&
s.CourseCode == course));
        }
        catch
        {
            return View();
        }
    }

    [HttpPost]
    public ActionResult ExamResultAsign(int studentCode, int courseCode,
FormCollection form)
    {
        try
        {
            var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode ==
studentCode && sc.CourseCode == courseCode);
            var course = _courses.GetSingleByCriteria(c => c.Code == courseCode);
            var student = _students.GetSingleByCriteria(s => s.StudentCode ==
studentCode);
            var examType = form.GetValue("SelectedExamType").ToString();
            var examResult = int.Parse(form.GetValue("ExamResultText").ToString());

            var tileNotification = new TileNotification { BackTitle = course.Title };

            switch (examType)
            {
                case "PartialExam":
                    studentCourse.PartialExamResult = examResult;
                    tileNotification.BackContent = String.Format("Examen Parcial: {0}",
examResult);
                    break;
                case "RecuperationExam":
                    studentCourse.RecuperationExamResult = examResult;
                    tileNotification.BackContent = String.Format("Examen Recuperatorio: {0}",
examResult);
                    break;
                case "FinalExam":
                    studentCourse.FinalExamResult = examResult;
```

```
                    tileNotification.BackContent = String.Format("Examen Final: {0}",
examResult);
                    break;
                default:
                    studentCourse.PracticWorkResult = examResult;
                    tileNotification.BackContent = String.Format("Presentación TPs: {0}",
examResult);
                    break;
            }

            _table.Add(studentCourse);
            var queueNotificationMessage = new QueueNotificationMessage
               {
                  Destinatary = student.DeviceUri,
                  Type = tileNotification.Type,
                  Payload = tileNotification.ToString()
               };

            _queueNotifications.AddMessage(queueNotificationMessage);

            return RedirectToAction("Index");
         }
         catch (Exception)
         {
            ViewData["ExamTypes"] =
                  new List<SelectListItem>
                     {
                        new SelectListItem{ Text = "Parcial", Value="PartialExam"},
                        new SelectListItem{ Text = "Recuperatorio",
Value="RecuperationExam"},
                        new SelectListItem{ Text = "Final", Value="FinalExam"},
                        new SelectListItem{ Text = "Presentación de Trabajos Practicos",
Value="PracticalWorkExam"},
                     }
               ;
            return View(_table.GetSingleByCriteria(s => s.StudentCode == studentCode &&
s.CourseCode == courseCode));
         }
      }

      public ActionResult AsignAssistedHours(int id, int course)
      {
         var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode == id &&
sc.CourseCode == course);
         return View(studentCourse);
      }

      [HttpPost]
      public ActionResult AsignAssistedHours(int studentCode, int courseCode,
FormCollection collection)
      {
         var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode ==
studentCode && sc.CourseCode == courseCode);
```

```
                try
                {
                    var hours = int.Parse(collection.GetValue("AddHours").ToString());
                    studentCourse.TotalHoursAssisted += hours;
                    _table.Add(studentCourse);
                    return RedirectToAction("Index");
                }
                catch (Exception)
                {

                    return View(studentCourse);
                }
            }
        }
    }
```

StudentController.cs

```
    using System.Web.Mvc;
    using TesinaMobileCloud.Data.Model;
    using TesinaMobileCloud.Data.Table;

    namespace TesinaMobileCloud.AdminSite.Controllers
    {
        public class StudentController : Controller
        {
            private readonly ITable<Student> _students;

            public StudentController(ITable<Student> students)
            {
                _students = students;
            }

            public ActionResult Index()
            {
                return View(_students.GetByCriteria(s => !string.IsNullOrEmpty(s.PartitionKey)));
            }

            public ActionResult Details(int id)
            {
                return View(_students.GetSingleByCriteria(s => s.StudentCode == id));
            }

            //
            // GET: /Course/Create

            public ActionResult Create()
            {
                return View(new Student());
            }

            //
            // POST: /Course/Create
```

```
        [HttpPost]
        public ActionResult Create(Student student)
        {
            try
            {
                _students.Add(student);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_students.GetSingleByCriteria(s => s.StudentCode == id));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _students.GetSingleByCriteria(s => s.StudentCode == id);
                UpdateModel(courseInCloud);
                _students.Add(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
                _students.Delete(_students.GetSingleByCriteria(c => c.PartitionKey == id &&
c.RowKey == row));
                return RedirectToAction("Index");
            }
            catch
            {
```

```
                return View();
            }
        }
    }
}
```

Views

_ViewStart.cshtml

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Career

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Nueva";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Dar de Alta Carrera</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}
```

```
<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Eliminar</h2>

<h3>Está seguro de eliminar esta Carrera?</h3>
<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
@using (Html.BeginForm())
{
    <p>
        <input type="submit" value="Eliminar" />
        |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<h2>Details</h2>

<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { id=Model.CareerCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Editar";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
```

```
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Career>

@{
    ViewBag.Title = "Carreras";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Carreras</h2>

<p>
    @Html.ActionLink("Nueva", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("CareerCode", "Código")
        </th>
        <th>
            @Html.Label("Title", "Título")
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { id = item.CareerCode }) |
            @Html.ActionLink("Detalles", "Details", new { id = item.CareerCode }) |
```

```
            @Html.ActionLink("Eliminar", "Delete", new { id = item.CareerCode })
        </td>
    </tr>
}

</table>
```

CareerCourseRelationship

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Asignar Cátedra a Carrera</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de la Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de la Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("YearInTheCareer", "Año de Cursada")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>


        <p>
            <input type="submit" value="Guardar" />
        </p>
```

```
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar Relación</h2>

<h3>Está seguro que desea eliminar esta relación?</h3>
<fieldset>
    <legend>Relación Cátedra-Carrera</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("YearInTheCareer", "Año en que se dicta")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.YearInTheCareer)
    </div>

</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Relación Cátedra-Carrera</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("YearInTheCareer", "Año en que se dicta")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.YearInTheCareer)
    </div>

</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { course = Model.CourseCode, career =
Model.CareerCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)
```

```
    <fieldset>
        <legend>Relación Cátedra-Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("YearInTheCareer", "Año en que se dicta")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.CareerCourseRelationship>

@{
    ViewBag.Title = "Asignar Cátedra a Carrera";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Asignación de Cátedras a Carreras</h2>

<p>
    @Html.ActionLink("Nueva asignación", "Create")
```

```
</p>
<table>
    <tr>
        <th>
            @Html.Label("CareerCode", "Código de Carrera")
        </th>
        <th>
            @Html.Label("CourseCode", "Código de Cátedra")
        </th>
        <th>
            @Html.Label("YearInTheCareer", "Año en que se dicta")
        </th>

        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CourseCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.YearInTheCareer)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { course = item.CourseCode, career =
item.CareerCode }) |
            @Html.ActionLink("Detalles", "Details",new { course = item.CourseCode, career =
item.CareerCode }) |
            @Html.ActionLink("Eliminar", "Delete", new { course = item.CourseCode, career =
item.CareerCode } )
        </td>
    </tr>
}

</table>
```

Course

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Nuevo";
}

<h2>Dar de Alta Cátedra</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)
```

```
    <fieldset>
        <legend>Cátedra</legend>

        <div class="editor-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Code)
            @Html.ValidationMessageFor(model => model.Code)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.Label("Professor", "Docente")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>
        <div class="editor-label">
            @Html.Label("FinalExamDate", "Fecha de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamDate)
            @Html.ValidationMessageFor(model => model.FinalExamDate)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExameDate)
            @Html.ValidationMessageFor(model => model.PartialExameDate)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkPresentationDate)
            @Html.ValidationMessageFor(model => model.PracticWorkPresentationDate)
        </div>
        <div class="editor-label">
            @Html.Label("RecuperationExameDate", "Fecha de Examen Recuperatorio")
        </div>
```

```
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExameDate)
            @Html.ValidationMessageFor(model => model.RecuperationExameDate)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>

        <div class="editor-label">
            @Html.Label("Scheduled", "Horario")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Scheduled)
            @Html.ValidationMessageFor(model => model.Scheduled)
        </div>

        <div class="editor-label">
            @Html.Label("Description", "Descripción de la Cátedra")
        </div>
        <div class="editor-field">
            @Html.TextAreaFor(model => model.Description)
            @Html.ValidationMessageFor(model => model.Description)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar</h2>

<h3>Está seguro que desea eliminar la Cátedra?</h3>
<fieldset>
    <legend>Course</legend>
```

```
    <div class="display-label">
        @Html.Label("Code", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Code)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>

    <div class="display-label">
        @Html.Label("Professor", "Docente")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Detalles";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Cátedra</legend>

    <div class="display-label">
        @Html.Label("Code", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Code)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
```

```
    </div>

    <div class="display-label">
        @Html.Label("Professor", "Docente")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
    <div class="editor-label">
        @Html.Label("FinalExamDate", "Fecha de Examen Final")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.FinalExamDate)
    </div>

    <div class="editor-label">
        @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.PartialExameDate)
    </div>

    <div class="editor-label">
        @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.PracticWorkPresentationDate)
    </div>
    <div class="editor-label">
        @Html.Label("RecuperationExame", "Fecha de Examen Recuperatorio")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.RecuperationExameDate)
    </div>
    <div class="editor-label">
        @Html.Label("TotalHoursOfClasses", "Total de Horas de Cursada")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.TotalHoursOfClasses)
    </div>

    <div class="editor-label">
        @Html.Label("Scheduled", "Horario")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.Scheduled)
    </div>

    <div class="editor-label">
        @Html.Label("Description", "Descripción de la Cátedra")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.Description)
```

```
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { id = Model.Code }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Cátedra</legend>

        <div class="editor-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Code)
            @Html.ValidationMessageFor(model => model.Code)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.Label("Professor", "Docente")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>

       <div class="editor-label">
            @Html.Label("FinalExamDate", "Fecha de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamDate)
            @Html.ValidationMessageFor(model => model.FinalExamDate)
```

```
        </div>

        <div class="editor-label">
            @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExameDate)
            @Html.ValidationMessageFor(model => model.PartialExameDate)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkPresentationDate)
            @Html.ValidationMessageFor(model => model.PracticWorkPresentationDate)
        </div>
        <div class="editor-label">
            @Html.Label("RecuperationExameDate", "Fecha de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExameDate)
            @Html.ValidationMessageFor(model => model.RecuperationExameDate)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>

        <div class="editor-label">
            @Html.Label("Scheduled", "Horario")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Scheduled)
            @Html.ValidationMessageFor(model => model.Scheduled)
        </div>

        <div class="editor-label">
            @Html.Label("Description", "Descripción de la Cátedra")
        </div>
        <div class="editor-field">
            @Html.TextAreaFor(model => model.Description)
            @Html.ValidationMessageFor(model => model.Description)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}
```

```
<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Course>

@{
    ViewBag.Title = "Cátedras";
}
<style type="text/css">
th {
    padding-right: 10px;
}
</style>
<h2>Cátedras</h2>

<p>
    @Html.ActionLink("Nueva", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("Code", "Código")
        </th>
        <th>
            @Html.Label("Title", "Título")
        </th>
        <th>
            @Html.Label("Professor", "Docente")
        </th>
        <th>
            @Html.Label("Scheduled", "Horario")
        </th>
        <th>
            @Html.Label("TotalHoursOfClasses", "Horas de Cursada")
        </th>
        <th></th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Code)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
```

```
            <td>
                @Html.DisplayFor(modelItem => item.Professor)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Scheduled)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursOfClasses)
            </td>
            <td>
                @Html.ActionLink("Editar", "Edit", new { id = item.Code  }) |
            @Html.ActionLink("Detalles", "Details", new { id = item.Code }) |
            @Html.ActionLink("Eliminar", "Delete", new { id = item.Code })
            </td>
        </tr>
    }

</table>
```

Home

Index.cshtml

```
@{
    ViewBag.Title = "Sitio de Administración - Tesina Carlos Rodrigo";
}
@section featured {
    <section class="featured">
        <div class="content-wrapper">
            <hgroup class="title">
                <h1>@ViewBag.Title.</h1>
            </hgroup>
            <p>
                Este sitio provee las funciones de administración y carga de datos necesarios
para el desarrollo de la tesina.
                La necesidad surge de simular los sistemas de asistencias y administración de
los alumnos, para solo dedicarse a
                la implementación del cliente Mobile.
            </p>
        </div>
    </section>
}
```

Shared

Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}
```

```
<hgroup class="title">
    <h1 class="error">Error.</h1>
    <h2 class="error">Se ha producido un error inesperado.</h2>
</hgroup>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width" />
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
                </div>
                <div class="float-right">
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("Cátedras", "Index", "Course")</li>
                            <li>@Html.ActionLink("Carreras", "Index", "Career")</li>
                            <li>@Html.ActionLink("Cátedras-Carreras", "Index",
"CareerCourseRelationship")</li>
                            <li>@Html.ActionLink("Estudiantes", "Index", "Student")</li>
                            <li>@Html.ActionLink("Estudiantes-Cátedras", "Index",
"StudentCodeRelationship")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
            </div>
        </footer>

        @Scripts.Render("~/bundles/jquery")
```

```
        @RenderSection("scripts", required: false)
    </body>
</html>
```

_LoginPartial.cshtml

```
@if (Request.IsAuthenticated) {
    <text>
        Hello, @Html.ActionLink(User.Identity.Name, "Manage", "Account", routeValues: null,
htmlAttributes: new { @class = "username", title = "Manage" })!
        @using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id =
"logoutForm" })) {
            @Html.AntiForgeryToken()
            <a href="javascript:document.getElementById('logoutForm').submit()">Log off</a>
        }
    </text>
} else {
    <ul>
        <li>@Html.ActionLink("Register", "Register", "Account", routeValues: null,
htmlAttributes: new { id = "registerLink" })</li>
        <li>@Html.ActionLink("Log in", "Login", "Account", routeValues: null, htmlAttributes:
new { id = "loginLink" })</li>
    </ul>
}
```

Student

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Create";
}

<h2>Nuevo Estudiante</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Estudiante</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
```

```
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("FirstName", "Nombres")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FirstName)
            @Html.ValidationMessageFor(model => model.FirstName)
        </div>

        <div class="editor-label">
            @Html.Label("LastName", "Apellido")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.LastName)
            @Html.ValidationMessageFor(model => model.LastName)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar Estudiante</h2>

<h3>Está seguro que desea eliminar este Estudiante?</h3>
<fieldset>
    <legend>Estudiante</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>
```

```
    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("FirstName", "Nombres")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FirstName)
    </div>

    <div class="display-label">
        @Html.Label("LastName", "Apellido")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.LastName)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Estudiante</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
```

```
    </div>

    <div class="display-label">
        @Html.Label("FirstName", "Nombres")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FirstName)
    </div>

    <div class="display-label">
        @Html.Label("LastName", "Apellido")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.LastName)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new {  id=Model.StudentCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Estudiante</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
```

```
            @Html.Label("FirstName", "Nombres")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FirstName)
            @Html.ValidationMessageFor(model => model.FirstName)
        </div>

        <div class="editor-label">
            @Html.Label("LastName", "Apellido")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.LastName)
            @Html.ValidationMessageFor(model => model.LastName)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Student>

@{
    ViewBag.Title = "Index";
}

<h2>Estudiantes</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("CareerCode", "Carrera")
        </th>
        <th>
            @Html.Label("StudentCode", "Matricula")
        </th>
        <th>
            @Html.Label("FirstName", "Nombre")
        </th>
        <th>
            @Html.Label("LastName", "Apellido")
```

```
            </th>
            <th></th>
        </tr>

    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.CareerCode)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.StudentCode)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.FirstName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LastName)
            </td>
            <td>
                @Html.ActionLink("Editar", "Edit", new {  id=item.StudentCode}) |
                @Html.ActionLink("Detalles", "Details", new { id=item.StudentCode }) |
                @Html.ActionLink("Eliminar", "Delete", new { id=item.StudentCode})
            </td>
        </tr>
    }

    </table>
```

StudentCodeRelationship

AsignAssistedHours.cshtml

```
    @model TesinaMobileCloud.Data.Model.StudentCourseRelationship

    @{
        ViewBag.Title = "Asignar Horas Cursadas";
        Layout = "_Layout";
    }

    <h2>Asignar Horas Cursadas</h2>
    <div>
    @using (Html.BeginForm()) {
        @Html.HiddenFor(model => model.StudentCode)
        @Html.HiddenFor(model => model.CourseCode)
        <div id="ActualAssitedHours">
            <label>Horas Asistidas a la fecha</label>
            @Html.DisplayFor(model => model.TotalHoursAssisted)
        </div>
        <div id="AddAssitedHours">
            @Html.TextBox("AddHours")
        </div>
        <div id="save">
            <input type="button" title="Guardar"/>
        </div>
        }
```

```
</div>
```

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>StudentCourseRelationship</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.StudentCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.CourseCode)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.PartialExamResult)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExamResult)
            @Html.ValidationMessageFor(model => model.PartialExamResult)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.FinalExamResult)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamResult)
            @Html.ValidationMessageFor(model => model.FinalExamResult)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.RecuperationExamResult)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExamResult)
```

```
            @Html.ValidationMessageFor(model => model.RecuperationExamResult)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.PracticWorkResult)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkResult)
            @Html.ValidationMessageFor(model => model.PracticWorkResult)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.TotalHoursAssisted)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursAssisted)
            @Html.ValidationMessageFor(model => model.TotalHoursAssisted)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.RowKey)
        </div>
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Create";
}

<h2>Asignación de Cátedras a Estudiantes</h2>
```

```
@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código de Estudiante")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        @*<div class="editor-label">
            @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExamResult)
            @Html.ValidationMessageFor(model => model.PartialExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("FinalExamResult", "Calificación de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamResult)
            @Html.ValidationMessageFor(model => model.FinalExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExamResult)
            @Html.ValidationMessageFor(model => model.RecuperationExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkResult)
            @Html.ValidationMessageFor(model => model.PracticWorkResult)
        </div>*@
```

```
        <div class="editor-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursAssisted)
            @Html.ValidationMessageFor(model => model.TotalHoursAssisted)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar Asignación</h2>

<h3>Está seguro que desea eliminar esta asignación?</h3>
<fieldset>
    <legend>Asignación de Estudiante a Cátedra</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código de Estudiante")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
    </div>
    <div class="display-field">
```

```
    @if (Model.PartialExamResult != 0)
    {
        @Html.DisplayFor(model => model.PartialExamResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("FinalExamResult", "Calificación de Examen Final")
</div>
<div class="display-field">
    @if (Model.FinalExamResult != 0)
    {
        @Html.DisplayFor(model => model.FinalExamResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
</div>
<div class="display-field">
    @if (Model.RecuperationExamResult != 0)
    {
        @Html.DisplayFor(model => model.RecuperationExamResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
</div>
<div class="display-field">
    @if (Model.PracticWorkResult != 0)
    {
        @Html.DisplayFor(model => model.PracticWorkResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
```

```
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.TotalHoursAssisted)
    </div>

</fieldset>
@using (Html.BeginForm())
{
    <p>
        <input type="submit" value="Eliminar" />
        |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Asignación Estudiante a Cátedra</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código de Estudiante")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.PartialExamResult)
    </div>

    <div class="display-label">
        @Html.Label("FinalExamResult", "Calificación de Examen Final")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FinalExamResult)
```

```
    </div>

    <div class="display-label">
        @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.RecuperationExamResult)
    </div>

    <div class="display-label">
        @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.PracticWorkResult)
    </div>

    <div class="display-label">
        @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.TotalHoursAssisted)
    </div>

</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new {  id=Model.StudentCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código de Estudiante")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>
```

```
        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.PartialExamResult)
            @if (Model.PartialExamResult != 0)
            {
                @Html.DisplayFor(model => model.PracticWorkResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "PartialExam" })
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("FinalExamResult", "Calificación de Examen Final")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.FinalExamResult)
            @if (Model.FinalExamResult != 0)
            {
                @Html.DisplayFor(model => model.FinalExamResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "FinalExam" })
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.RecuperationExamResult)
            @if (Model.RecuperationExamResult != 0)
            {
                @Html.DisplayFor(model => model.RecuperationExamResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "RecuperationExam" })
            }
            else
```

```
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.PracticWorkResult)
            @if (Model.PracticWorkResult != 0)
            {
                @Html.DisplayFor(model => model.PracticWorkResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "PracticWorkResult" })
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="editor-field">
            @Html.DisplayFor(model => model.TotalHoursAssisted)
            @Html.ActionLink("Asignar Horas Cursadas", "AsignAssistedHours", new { id =
Model.StudentCode, course = Model.CourseCode })
        </div>


        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

ExamResultAsign.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Asignar Calificación de Examen";
}
```

```
<h2>Asignar Calificación de Examen</h2>
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>
        @Html.HiddenFor(model => model.StudentCode)
        @Html.HiddenFor(model => model.CourseCode)
        <div class="editor-label">
            @Html.Label("ExamType", "Examen")
        </div>
        <div class="editor-field">
            @Html.DropDownList("SelectedExamType",
(List<SelectListItem>)ViewData["ExamTypes"])
        </div>

        <div class="editor-label">
            @Html.Label("ExamResult", "Calificación")
        </div>
        <div class="editor-field">
            @Html.TextBox("ExamResultText")
        </div>
    </fieldset>
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.StudentCourseRelationship>

@{
    ViewBag.Title = "Index";
}

<h2>Estado del Estudiante en una Cátedra</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("StudentCode", "Matricula del Estudiante")
        </th>
        <th>
            @Html.Label("CourseCode", "Código de la Cátedra")
        </th>
        <th>
            @Html.Label("PartialExamResult", "Resultado de Examen Parcial")
        </th>
        <th>
            @Html.Label("FinalExamResult", "Resultado de Examen Final")
        </th>
        <th>
```

```
            @Html.Label("RecuperationExamResult", "Resultado de Examen Recuperatorio")
        </th>
        <th>
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </th>
        <th>
            @Html.Label("TotalHoursAssisted", "Total de Horas Asistidas")
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.StudentCode)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.CourseCode)
            </td>
            <td>
                @if (item.PartialExamResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.PartialExamResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @if (item.FinalExamResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.FinalExamResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @if (item.RecuperationExamResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.RecuperationExamResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @if (item.PracticWorkResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.PracticWorkResult)
```

```
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursAssisted)
            </td>

            <td>
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
item.StudentCode, course = item.CourseCode })
                @Html.ActionLink("Asignar Horas Cursadas", "AsignAssistedHours", new { id =
item.StudentCode, courseCode = item.CourseCode })
                @Html.ActionLink("Editar", "Edit", new { id = item.StudentCode, courseCode =
item.CourseCode }) |
                @Html.ActionLink("Detalles", "Details", new { id = item.StudentCode, courseCode
= item.CourseCode }) |
                @Html.ActionLink("Eliminar", "Delete", new { id = item.StudentCode, courseCode
= item.CourseCode })
            </td>
        </tr>
    }

</table>
```

TesinaMobileCloud.Data

CloudStorageConfiguration.cs

```
using System;
using System.Configuration;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.Data
{
    public class CloudStorageConfiguration
    {
        public static CloudStorageAccount GetCloudAccount(string
cloudStorageConnectionStringName)
        {
            try
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
            catch (InvalidOperationException)
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
        }
```

```
        private static CloudStorageAccount SetConfigurationAndGetAccount(string
cloudStorageConnectionStringName)
        {
            SetConfigurationSettingsPublisher();
            return
CloudStorageAccount.FromConfigurationSetting(cloudStorageConnectionStringName);
        }

        private static void SetConfigurationSettingsPublisher()
        {
            CloudStorageAccount.SetConfigurationSettingPublisher((configName,
configSettingPublisher) =>
            {
                var configValue = ConfigurationManager.AppSettings[configName];
                if (RoleEnvironment.IsAvailable)
                    configValue = RoleEnvironment.GetConfigurationSettingValue(configName);
                configSettingPublisher(configValue);
            });
        }

    }
}
```

DTO

Attendance.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Attendance
    {
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public DateTime TotalHoursOfClasses { get; set; }
        [DataMember]
        public int ActualPercentage { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Career
    {
        [DataMember]
        public string Code { get; set; }
```

```
        [DataMember]
        public string Title { get; set; }
    }
}
```

Course.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Course
    {
        [DataMember]
        public int Code { get; set; }
        [DataMember]
        public string Title { get; set; }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public int YearOfCareer { get; set; }
        [DataMember]
        public DateTime PartialExamDate { get; set; }
        [DataMember]
        public DateTime FinalExamDate { get; set; }
        [DataMember]
        public DateTime RecuperationExameDate { get; set; }
        [DataMember]
        public DateTime PracticWorkPresentationDate { get; set; }
        [DataMember]
        public int TotalHoursOfClasses { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Scheduled { get; set; }
    }
}
```

Student.cs

```
using System.Collections.Generic;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Student
    {
        public Student()
        {
            Courses = new List<StudentCourse>();
        }
```

```
        [DataMember]
        public IList<StudentCourse> Courses { get; set; }
        [DataMember]
        public Career Career { get; set; }
        [DataMember]
        public string FirstName { get; set; }
        [DataMember]
        public string LastName { get; set; }
        [DataMember]
        public int StudentCode { get; set; }
        [DataMember]
        public int CareerCode { get; set; }
    }
}
```

StudentCourse.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class StudentCourse
    {
        [DataMember]
        public double PartialExamResult { get; set; }
        [DataMember]
        public double FinalExamResult { get; set; }
        [DataMember]
        public double RecuperationExamResult { get; set; }
        [DataMember]
        public double PracticWorkResult { get; set; }
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public Course Course { get; set; }
        [DataMember]
        public bool PartialExamInscripted { get; set; }
        [DataMember]
        public bool FinalExamInscripted { get; set; }

        public double Attendance()
        {
            return (TotalHoursAssisted * 100) / Course.TotalHoursOfClasses;
            //if (result > 75)
            //    return 3;
            //if (result < 75 && result > 50)
            //    return 2;
            //return 3;
        }
    }
}
```

Helpers

JsonSerializationHelper.cs

```
using System.IO;
using System.Runtime.Serialization.Json;
using System.Text;

namespace TesinaMobileCloud.Data.Helpers
{
    public class JsonSerializationHelper
    {
        public static string Serialize<T>(T entity)
        {
            var ser = new DataContractJsonSerializer(typeof(T));
            var ms = new MemoryStream();
            ser.WriteObject(ms, entity);
            var jsonString = Encoding.UTF8.GetString(ms.ToArray());
            ms.Close();
            return jsonString;
        }

        public static T JsonDeserialize<T>(string jsonString)
        {
            var ser = new DataContractJsonSerializer(typeof(T));
            var ms = new MemoryStream(Encoding.UTF8.GetBytes(jsonString));
            T obj = (T)ser.ReadObject(ms);
            return obj;
        }

    }
}
```

Model

AzureEntity.cs

```
using System;
using System.Data.Services.Common;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataServiceEntity]
    [DataContract]
    public class AzureEntity
    {
        public string RowKey { get; set; }
        public string PartitionKey { get; set; }
        public DateTime Timestamp { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;
```

```
namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Career : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return RowKey; }
            set { RowKey = value; }
        }

        public Career()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Career(int careerCode, string title)
        {
            CareerCode = careerCode;
            Title = title;
        }


    }
}
```

CareerCourseRelationship.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class CareerCourseRelationship : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
```

```
        [DataMember]
        public int YearInTheCareer { get; set; }

        public CareerCourseRelationship()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public CareerCourseRelationship(int careerCode, int courseCode)
        {
            CareerCode = careerCode;
            CourseCode = courseCode;
        }
    }
}
```

Course.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Course : AzureEntity
    {
        [DataMember]
        public int Code
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return string.IsNullOrEmpty(RowKey) ? string.Empty : RowKey; }
            set { RowKey = value; }
        }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public DateTime FinalExamDate { get; set; }
        [DataMember]
        public DateTime PartialExameDate { get; set; }
        [DataMember]
        public DateTime PracticWorkPresentationDate { get; set; }
        [DataMember]
        public DateTime RecuperationExameDate { get; set; }
        [DataMember]
        public int TotalHoursOfClasses { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Scheduled { get; set; }
```

```
        public Course()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Course(int courseCode, string name)
        {
            Code = courseCode;
            Title = name;
        }
    }
}
```

Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Student : AzureEntity
    {
        [DataMember]
        public int StudentCode
        {
            get { return int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CareerCode
        {
            get { return int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public string FirstName { get; set; }
        [DataMember]
        public string LastName { get; set; }
        [DataMember]
        public Uri DeviceUri { get; set; }

        public Student()
        {
            PartitionKey = "0";
            RowKey = "0";
        }
    }
```

```
}
```

StudentCourseRelationship.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class StudentCourseRelationship : AzureEntity
    {
        [DataMember]
        public int StudentCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public double PartialExamResult { get; set; }
        [DataMember]
        public double FinalExamResult { get; set; }
        [DataMember]
        public double RecuperationExamResult { get; set; }
        [DataMember]
        public double PracticWorkResult { get; set; }
        [DataMember]
        public int TotalHoursAssisted { get; set; }

        public StudentCourseRelationship()
        {
            PartitionKey = "0";
            RowKey = "0";
        }
    }
}
```

Queue

IQueue.cs

```
namespace TesinaMobileCloud.Data.Queue
{
    public interface IQueue<T>
    {
        void AddMessage(T message);
        T GetMessage();
        bool HasMessage { get; }
    }
```

```
}
```

NotificationQueue.cs

```
using System;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;
using TesinaMobileCloud.Data.Helpers;

namespace TesinaMobileCloud.Data.Queue
{
    public class NotificationQueue<T> : IQueue<T>
    {
        private CloudQueue _queue;
        private string _queueName = "notificationqueue";

        public NotificationQueue(CloudStorageAccount cloudStorageAccount)
        {
            var cloudQueueClient = cloudStorageAccount.CreateCloudQueueClient();
            _queue = cloudQueueClient.GetQueueReference(_queueName);
            _queue.CreateIfNotExist();
        }

        public void AddMessage(T message)
        {
            var messageString = JsonSerializationHelper.Serialize<T>(message);
            _queue.AddMessage(new CloudQueueMessage(messageString));
        }

        public T GetMessage()
        {
            var decodedMessage = _queue.GetMessage();
            return JsonSerializationHelper.JsonDeserialize<T>(decodedMessage.AsString);
        }

        public bool HasMessage {
            get { return _queue.ApproximateMessageCount != null; }
        }
    }
}
```

QueueNotificationMessage.cs

```
using System;
using Microsoft.WindowsAzure.StorageClient;
using TesinaMobileCloud.Data.Helpers;

namespace TesinaMobileCloud.Data.Queue
{
    public class QueueNotificationMessage
    {
        public Uri Destinatary { get; set; }

        public string Type { get; set; }
```

```
        public string Payload { get; set; }
    }
}
```

Repository

CareerCourseRelationshipRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CareerCourseRelationshipRepository :
ICourseCareerRelationshipRepository
    {
        private readonly ITable<CareerCourseRelationship> _table;

        public CareerCourseRelationshipRepository(ITable<CareerCourseRelationship> table)
        {
            _table = table;
        }

        public int GetCareerYearOfCourseByCareer(int careerCode, int courseCode)
        {
            return
                _table.GetSingleByCriteria(
                    relationship => relationship.CareerCode == careerCode &&
relationship.CourseCode == courseCode).
                    YearInTheCareer;
        }
    }
}
```

CareerRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CareerRepository : ICareerRepository
    {
        private readonly ITable<Career> _careerTable;

        public CareerRepository(ITable<Career> careerTable)
        {
            _careerTable = careerTable;
        }

        public Career GetCareer(string careerCode)
        {
```

```
            return _careerTable.GetSingleByCriteria(career => career.CareerCode ==
        int.Parse(careerCode));
        }
    }
```

CourseRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CourseRepository : ICourseRepository
    {
        private readonly ITable<Course> _courseTable;
        private readonly ITable<CareerCourseRelationship> _careerCourseTable;

        public CourseRepository(ITable<Course> courseTable,
        ITable<CareerCourseRelationship> careerCourseTable)
        {
            _courseTable = courseTable;
            _careerCourseTable = careerCourseTable;
        }

        public IList<Course> GetCoursesByCareer(string careerCode)
        {
            var courses = _careerCourseTable.GetByCriteria(c => c.CareerCode ==
        int.Parse(careerCode));
            return courses.Select(careerCourseRelationship =>
        _courseTable.GetSingleByCriteria(c => c.Code ==
        careerCourseRelationship.CourseCode)).ToList();
        }

        public IList<Course> GetAllCourses()
        {
            return _courseTable.GetByCriteria(c => !string.IsNullOrEmpty(c.PartitionKey));
        }

        public void AddCourse(Course course)
        {
            _courseTable.Add(course);
        }

        public Course GetSingleCourse(int courseCode)
        {
            return _courseTable.GetSingleByCriteria(c => c.Code == courseCode);
        }

        public void DeleteCourse(int partitionKey)
        {
            _courseTable.Delete(_courseTable.GetSingleByCriteria(c => c.Code ==
        partitionKey));
```

```
        }
    }
}
```

StudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class StudentCourseRepository : IStudentCourseRepository
    {
        private readonly ITable<StudentCourseRelationship> _studentCourseTable;

        public StudentCourseRepository(ITable<StudentCourseRelationship>
studentCourseTable)
        {
            _studentCourseTable = studentCourseTable;
        }

        public IEnumerable<StudentCourseRelationship> RetriveCoursesOfStudent(string
studentCode)
        {
            return _studentCourseTable.GetByCriteria(s => s.StudentCode ==
int.Parse(studentCode));
        }

        public StudentCourseRelationship RetriveStudentCourseByCode(string studentCode,
string courseCode)
        {
            return _studentCourseTable.GetSingleByCriteria(sc => sc.StudentCode ==
int.Parse(studentCode) && sc.CourseCode == int.Parse(courseCode));
        }
    }
}
```

StudentRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class StudentRepository : IStudentRepository
    {
        private readonly ITable<Student> _studentTable;

        public StudentRepository(ITable<Student> studentTable)
        {
            _studentTable = studentTable;
        }
```

```
        public Student GetStudentByCareerCodeAndStudentCode(string studentCode, string
    careerCode)
        {
            return _studentTable.GetSingleByCriteria(s => s.StudentCode ==
    int.Parse(studentCode) &&
                                                    s.CareerCode == int.Parse(careerCode));
        }
    }
}
```

Interfaces

ICareerRepository.cs

```
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICareerRepository
    {
        Career GetCareer(string careerCode);
    }
}
```

ICourseCareerRelationshipRepository.cs

```
namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseCareerRelationshipRepository
    {
        int GetCareerYearOfCourseByCareer(int careerCode, int courseCode);
    }
}
```

ICourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseRepository
    {
        IList<Course> GetCoursesByCareer(string careerCode);
        IList<Course> GetAllCourses();
        void AddCourse(Course course);
        Course GetSingleCourse(int courseCode);
    }
}
```

IStudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
```

```
{
    public interface IStudentCourseRepository
    {
        IEnumerable<StudentCourseRelationship> RetriveCoursesOfStudent(string
studentCode);
        StudentCourseRelationship RetriveStudentCourseByCode(string studentCode, string
courseCode);
    }
}
```

IStudentRepository.cs

```
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface IStudentRepository
    {
        Student GetStudentByCareerCodeAndStudentCode(string studentCode, string
careerCode);
    }
}
```

Table

AzureTable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;

namespace TesinaMobileCloud.Data.Table
{
    public class AzureTable<T> : ITable<T>
    {
        private readonly TableServiceContext _context;
        private readonly string _tableName;
        private IQueryable<T> Query { get; set; }

        public AzureTable(CloudStorageAccount cloudStorageAccount)
        {
            _tableName = typeof (T).Name;

            var table = new CloudTableClient(cloudStorageAccount.TableEndpoint.ToString(),
cloudStorageAccount.Credentials);
            _context = table.GetDataServiceContext();
            table.CreateTableIfNotExist(_tableName);
            Query = _context.CreateQuery<T>(_tableName).AsTableServiceQuery();
        }

        public IList<T> GetByCriteria(Func<T, bool> func)
        {
            return Query.Where(func).ToList();
```

```
        }

        public T GetSingleByCriteria(Func<T, bool> func)
        {
            return Query.SingleOrDefault(func);
        }

        public void Add(T entity)
        {
            try
            {
                _context.AddObject(_tableName, entity);
            }
            catch (InvalidOperationException)
            {
                _context.UpdateObject(entity);
            }
            _context.SaveChanges();

        }

        public void Delete(T entity)
        {
            _context.DeleteObject(entity);
            _context.SaveChanges();
        }
    }
}
```

ITable.cs

```
using System;
using System.Collections.Generic;

namespace TesinaMobileCloud.Data.Table
{
    public interface ITable<T>
    {
        IList<T> GetByCriteria(Func<T, bool> func);
        T GetSingleByCriteria(Func<T, bool> func);
        void Add(T entity);
        void Delete(T entity);
    }
}
```

TesinaMobileCloud.Data.Test

CourseCareerRelationshipRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;
```

```
namespace TesinaMobileCloud.Data.Test
{
   [TestClass]
   public class CourseCareerRelationshipRepositoryFixture
   {
      private Mock<ITable<CareerCourseRelationship>> _mockCourseCareerRelationship;

      [TestInitialize]
      public void Setup()
      {
         _mockCourseCareerRelationship = new
Mock<ITable<CareerCourseRelationship>>();
      }

      [TestMethod]
      public void GetCareerYearOfCourseByCareer()
      {
         var career = 502;
         var course = 120;
         _mockCourseCareerRelationship.Setup(
            m => m.GetSingleByCriteria(It.IsAny<Func<CareerCourseRelationship,
bool>>())).Returns(new CareerCourseRelationship
                                                                  {
                                                                     CareerCode = career,
                                                                     CourseCode =
course,
                                                                     YearInTheCareer = 1
                                                                  });

         var sut = new
CareerCourseRelationshipRepository(_mockCourseCareerRelationship.Object);
         var result = sut.GetCareerYearOfCourseByCareer(career, course);

         Assert.AreEqual(1, result);


      }
   }
}
```

CourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
   [TestClass]
   public class CourseRepositoryFixture
   {
```

```
        private Mock<ITable<Course>> _mockCourse;
        private Mock<ITable<CareerCourseRelationship>> _mockCareerCourse;

        [TestInitialize]
        public void Setup()
        {
            _mockCourse = new Mock<ITable<Course>>();
            _mockCareerCourse = new Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCoursesByCareer()
        {
            const int careerCode = 502;
            _mockCareerCourse.Setup(m =>
m.GetByCriteria(It.IsAny<Func<CareerCourseRelationship, bool>>())).Returns(new
List<CareerCourseRelationship>
                                                                        {
                                                                            new
CareerCourseRelationship(502, 123),
                                                                            new
CareerCourseRelationship(502, 124),
                                                                        });
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new Course(123, "Programación II"));
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetCoursesByCareer(careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 2);
        }

        [TestMethod]
        public void GetAllCourses()
        {
            _mockCourse.Setup(m => m.GetByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new List<Course>{ new Course(123, "Programación II")});
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 1);
        }

        [TestMethod]
        public void DeleteCourse()
        {
            var course = new Course(122, "Programación I");
            var courses = new List<Course>
                          {
```

```
                    new Course(123, "Programación II"),
                    course
                };
            _mockCourse.Setup(m => m.Delete(course)).Callback(() =>
courses.Remove(course));
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(course);

            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            sut.DeleteCourse(122);

            Assert.AreEqual(1, courses.Count);
        }

        [TestMethod]
        public void AddCourse()
        {
            var course = new Course(123, "Programación II");
            _mockCourse.Setup(m => m.Add(It.IsAny<Course>())).Verifiable();

            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);
            sut.AddCourse(course);
            _mockCourse.Verify();
        }
    }
}
```

StudentCourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class StudentCourseRepositoryFixture
    {
        private Mock<ITable<StudentCourseRelationship>> _studentCourseTable;

        [TestInitialize]
        public void Setup()
        {
            _studentCourseTable = new Mock<ITable<StudentCourseRelationship>>();
        }
        [TestMethod]
        public void RetriveCoursesOfStudent()
```

```
        {
            _studentCourseTable.Setup(m =>
m.GetByCriteria(It.IsAny<Func<StudentCourseRelationship, bool>>()))
                .Returns(new List<StudentCourseRelationship>());
            var sut = new StudentCourseRepository(_studentCourseTable.Object);

            var result = sut.RetriveCoursesOfStudent("10004");

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result,
typeof(IEnumerable<StudentCourseRelationship>));
        }

        [TestMethod]
        public void RetriveStudentCourseByCode()
        {
            const string studentCode = "2";
            const string courseCode = "3";
            _studentCourseTable.Setup(m =>
m.GetSingleByCriteria(It.IsAny<Func<StudentCourseRelationship, bool>>()))
                .Returns(new StudentCourseRelationship
                {
                    StudentCode = int.Parse(studentCode),
                    CourseCode = int.Parse(courseCode)
                });
            var sut = new StudentCourseRepository(_studentCourseTable.Object);

            var result = sut.RetriveStudentCourseByCode(studentCode, courseCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof (StudentCourseRelationship));
            Assert.AreEqual(studentCode, result.StudentCode);
            Assert.AreEqual(courseCode, result.CourseCode);
        }
    }
}
```

StudentRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class StudentRepositoryFixture
    {
        private Mock<ITable<Student>> _mockStudentTable;

        [TestInitialize]
```

```
        public void Setup()
        {
            _mockStudentTable = new Mock<ITable<Student>>();
        }

        [TestMethod]
        public void GetStudentByCareerCodeAndStudentCode()
        {
            const int studentCode = 10004;
            const int careerCode = 502;
            _mockStudentTable.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Student,
bool>>()))
                             .Returns(new Student
                                      {
                                          CareerCode = careerCode,
                                          StudentCode = studentCode
                                      });

            var sut = new StudentRepository(_mockStudentTable.Object);

            var result = sut.GetStudentByCareerCodeAndStudentCode(studentCode.ToString(),
careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(10004, result.StudentCode);
            Assert.AreEqual(502, result.CareerCode);
        }
    }
}
```

UnitTest1.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class QueueTest
    {
        [TestMethod]
        public void TestMethod1()
        {
        }
    }
}
```

TesinaMobileCloud.Notifications

TileNotification.cs

```
namespace TesinaMobileCloud.Notifications
{
    public class TileNotification
```

```
    {
        public string BackTitle { get; set; }

        public string BackContent { get; set; }

        public string Type
        {
            get { return "token"; }
        }
    }
}
```

TesinaMobileCloud.Phone.Agents

ScheduledAgent.cs

```
using System;
using System.Windows;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Data.Repositories;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Phone.Services.Support;
using Microsoft.Phone.Shell;
using System.Linq;

namespace TesinaMobileCloud.Phone.Agents
{
    public class ScheduledAgent : ScheduledTaskAgent
    {
        private static volatile bool _classInitialized;
        private readonly ICloudService _cloudService;
        private IStudentInformationManager _studentInformationManager;

        /// <remarks>
        /// ScheduledAgent constructor, initializes the UnhandledException handler
        /// </remarks>
        public ScheduledAgent()
        {
            if (!_classInitialized)
            {
                _classInitialized = true;
                // Subscribe to the managed exception handler
                Deployment.Current.Dispatcher.BeginInvoke(delegate
                {
                    Application.Current.UnhandledException +=
ScheduledAgent_UnhandledException;
                });
            }

            _cloudService = new CloudService();
```

```
            _studentInformationManager = new StudentInformationManager(new
StudentRepository(),
                                        new StudentCourseRepository(),
                                        new ScheduleActionServiceAdapter());
        }

        /// Code to execute on Unhandled Exceptions
        private void ScheduledAgent_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        /// <summary>
        /// Agent that runs a scheduled task
        /// </summary>
        /// <param name="task">
        /// The invoked task
        /// </param>
        /// <remarks>
        /// This method is called when a periodic or resource intensive task is invoked
        /// </remarks>
        protected override void OnInvoke(ScheduledTask task)
        {
            if (task is PeriodicTask)
            {
                PerformSyncronizationTask();
            }
        }
        private void PerformSyncronizationTask()
        {
            var syncService = new SyncronizationService(_cloudService,
_studentInformationManager);

            syncService.SyncCourses()
                    .ObserveOnDispatcher()
                    .Subscribe(SyncCompleted, SyncFailed);
        }

        private void SyncCompleted(TaskSummary result)
        {
            DisplayShellNotification(new StandardTileData
                                {
                                    Title = "SIA",
                                    BackContent = "Información Actualizada"
                                });
            NotifyComplete();
        }

        private void SyncFailed(Exception exception)
```

```
            {
                DisplayShellNotification(new StandardTileData
                                         {
                                             Title = "SIA",
                                             BackContent = "Error al sincronizar datos"
                                         });
                Abort();
            }

            private static void DisplayShellNotification(ShellTileData tileData)
            {
                var tile = ShellTile.ActiveTiles.First();
                tile.Update(tileData);
            }
        }
    }
```

TesinaMobileCloud.Phone.Client

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone" xmlns:sys="clr-
namespace:System;assembly=mscorlib"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.ViewModel" mc:Ignorable="d">
    <!--Application Resources-->
    <Application.Resources>
        <ResourceDictionary>
            <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
            <Style TargetType="TextBlock" x:Key="TextBlockHeader">
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="32" />
                <Setter Property="TextWrapping" Value="Wrap"/>
                <Setter Property="FontFamily" Value="Segoe WP Semibold"/>
                <Setter Property="Margin" Value="0,0,0,5"/>
            </Style>
            <Style TargetType="TextBlock" x:Key="TextBlockContent">
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="25" />
                <Setter Property="TextWrapping" Value="Wrap"/>
            </Style>
            <SolidColorBrush x:Key="PhoneControlBackgroundBrush" Color="#FF0083B6"/>
            <Color x:Key="ApplicationBarBrush">#FF0F5CE5</Color>
        </ResourceDictionary>
    </Application.Resources>
    <Application.ApplicationLifetimeObjects>
        <!--Required object that handles lifetime events for the application-->
        <shell:PhoneApplicationService Launching="Application_Launching"
                                       Closing="Application_Closing"
```

```
                                Activated="Application_Activated"
                                Deactivated="Application_Deactivated" />
        </Application.ApplicationLifetimeObjects>
    </Application>
```

App.xaml.cs

```
    using System.Windows;
    using System.Windows.Navigation;
    using Microsoft.Phone.Controls;
    using Microsoft.Phone.Shell;
    using TesinaMobileCloud.Phone.Client.ViewModel;

    namespace TesinaMobileCloud.Phone.Client
    {
        public partial class App : Application
        {
            /// <summary>
            /// Provides easy access to the root frame of the Phone Application.
            /// </summary>
            /// <returns>The root frame of the Phone Application.</returns>
            public PhoneApplicationFrame RootFrame { get; private set; }

            protected ViewModelLocator Locator
            {
                get { return (ViewModelLocator)Resources["Locator"]; }
            }

            /// <summary>
            /// Constructor for the Application object.
            /// </summary>
            public App()
            {
                TiltEffect.TiltableItems.Add(typeof(CourseResumeViewModel));
                // Global handler for uncaught exceptions.
                UnhandledException += Application_UnhandledException;

                // Standard Silverlight initialization
                InitializeComponent();

                // Phone-specific initialization
                InitializePhoneApplication();

                // Show graphics profiling information while debugging.
                if (System.Diagnostics.Debugger.IsAttached)
                {
                    // Display the current frame rate counters.
                    Application.Current.Host.Settings.EnableFrameRateCounter = true;

                    // Show the areas of the app that are being redrawn in each frame.
                    //Application.Current.Host.Settings.EnableRedrawRegions = true;

                    // Enable non-production analysis visualization mode,
                    // which shows areas of a page that are handed off to GPU with a colored
    overlay.
```

```
            //Application.Current.Host.Settings.EnableCacheVisualization = true;

            // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
            // application's PhoneApplicationService object to Disabled.
            // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
            // and consume battery power when the user is not using the phone.
            PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
        }

    }

    // Code to execute when the application is launching (eg, from Start)
    // This code will not execute when the application is reactivated
    private void Application_Launching(object sender, LaunchingEventArgs e)
    {
        Locator.ScheduleActions.RemovePeriodicTask();
    }

    // Code to execute when the application is activated (brought to foreground)
    // This code will not execute when the application is first launched
    private void Application_Activated(object sender, ActivatedEventArgs e)
    {

    }

    // Code to execute when the application is deactivated (sent to background)
    // This code will not execute when the application is closing
    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
    {
        // Ensure that required application state is persisted here.
    }

    // Code to execute when the application is closing (eg, user hit Back)
    // This code will not execute when the application is deactivated
    private void Application_Closing(object sender, ClosingEventArgs e)
    {
        Locator.ScheduleActions.AddPeriodicTask();
    }

    // Code to execute if a navigation fails
    private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
    {
        if (System.Diagnostics.Debugger.IsAttached)
        {
            // A navigation has failed; break into the debugger
            System.Diagnostics.Debugger.Break();
        }
    }

    // Code to execute on Unhandled Exceptions
```

```
        private void Application_UnhandledException(object sender,
    ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new TransitionFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
    e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

MainPage.xaml

```
<client:PhonePage
    x:Class="TesinaMobileCloud.Phone.Client.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:View="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:client="clr-namespace:TesinaMobileCloud.Phone.Client"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="800"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"
    Orientation="Portrait"
    shell:SystemTray.IsVisible="False">
    <toolkit:TransitionService.NavigationInTransition>
        <toolkit:NavigationInTransition>
            <toolkit:NavigationInTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardIn"/>
            </toolkit:NavigationInTransition.Backward>
            <toolkit:NavigationInTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardIn"/>
            </toolkit:NavigationInTransition.Forward>
        </toolkit:NavigationInTransition>
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
        <toolkit:NavigationOutTransition>
            <toolkit:NavigationOutTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardOut"/>
            </toolkit:NavigationOutTransition.Backward>
            <toolkit:NavigationOutTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardOut"/>
            </toolkit:NavigationOutTransition.Forward>
        </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

            <client:PhonePage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Mode="Minimized"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="0.35">
            <shell:ApplicationBar.MenuItems>
                        <shell:ApplicationBarMenuItem Text="Configuraciones" IsEnabled="True"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </client:PhonePage.ApplicationBar>

            <Grid x:Name="LayoutRoot" Background="Transparent">
        <controls:Panorama Title="Sia" Background="{StaticResource
PhoneControlBackgroundBrush}">
            <controls:PanoramaItem Header="Cátedras">
                <View:CoursesView/>
            </controls:PanoramaItem>
        </controls:Panorama>
    </Grid>
```

```
</client:PhonePage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client
{
    public partial class MainPage : PhonePage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

PhonePage.cs

```
using System.Windows.Navigation;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client
{
    public class PhonePage : PhoneApplicationPage
    {
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            var dataContext = DataContext as ViewModel.ViewModel;
            if (e.NavigationMode == NavigationMode.Back) return;

            if (dataContext != null)
                dataContext.OnNavigateTo(NavigationContext.QueryString);

            base.OnNavigatedTo(e);
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            var dataContext = DataContext as ViewModel.ViewModel;
            if (e.NavigationMode == NavigationMode.Forward) return;

            if (dataContext != null)
                dataContext.OnNavigateFrom(NavigationContext.QueryString);
```

```
            base.OnNavigatedFrom(e);
        }
    }
}
```

View

CourseDescriptionView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client"
x:Class="TesinaMobileCloud.Phone.Client.CourseDescriptionView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    DataContext="{Binding CourseDescription, Source={StaticResource Locator}}"
    d:DesignHeight="607" d:DesignWidth="456">

    <Grid x:Name="LayoutRoot">
        <ListBox HorizontalAlignment="Left" Height="607" VerticalAlignment="Top"
Width="456">
            <TextBlock x:Name="Description" Margin="0,0,0,10" Style="{StaticResource
TextBlockContent}" Text="{Binding Description}" Width="456"/>
            <TextBlock x:Name="Professor" Style="{StaticResource TextBlockHeader}"
Text="Docente"/>
            <TextBlock x:Name="ProfessorName" Text="{Binding Professor}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="DayAndTime" Style="{StaticResource TextBlockHeader}"
Text="Horario"/>
            <TextBlock x:Name="DayAndTimeContent" Text="{Binding Scheduled}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PartialExamDate" Style="{StaticResource TextBlockHeader}"
Text="Examen Parcial"/>
            <TextBlock x:Name="PartialExamDateContent" Text="{Binding PartialExamDate}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="RecuperationExameDate" Text="Examen Recuperatorio"
Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="RecuperationExameDateContent" Text="{Binding
RecuperationExamDate}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="FinalExamDate" Text="Examen Final" Style="{StaticResource
TextBlockHeader}"/>
            <TextBlock x:Name="FinalExamDateContent" Text="{Binding FinalExamDate}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PracticWorkExameDate" Text="Trabajos Practicos"
Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="PracticWorkExameDateContent" Text="{Binding
PracticWorkExamDate}" Style="{StaticResource TextBlockContent}"/>
```

```
            <TextBlock x:Name="TotalHoursOfClasses" Text="Total de horas de cursada"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="TotalHoursOfClassesContent" Text="{Binding
        TotalHoursOfClasses}" Style="{StaticResource TextBlockContent}"/>
        </ListBox>
      </Grid>
    </UserControl>
```

CourseDescriptionView.xaml.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Windows;
    using System.Windows.Controls;
    using System.Windows.Navigation;
    using Microsoft.Phone.Controls;
    using Microsoft.Phone.Shell;

    namespace TesinaMobileCloud.Phone.Client
    {
       public partial class CourseDescriptionView : UserControl
       {
          public CourseDescriptionView()
          {
             InitializeComponent();
          }
       }
    }
```

CourseResumeView.xaml

```
    <UserControl
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
            xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
            xmlns:i="clr-
    namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
    xmlns:ec="clr-
    namespace:Microsoft.Expression.Interactivity.Core;assembly=Microsoft.Expression.Interact
    ions" xmlns:Command="clr-
    namespace:GalaSoft.MvvmLight.Command;assembly=GalaSoft.MvvmLight.Extras.WP71"
    x:Class="TesinaMobileCloud.Phone.Client.View.CourseResumeView"
            mc:Ignorable="d"
            FontFamily="{StaticResource PhoneFontFamilyNormal}"
            FontSize="{StaticResource PhoneFontSizeNormal}"
            Foreground="{StaticResource PhoneForegroundBrush}"
            d:DesignHeight="85" d:DesignWidth="432" RenderTransformOrigin="0.5,0.5">

            <Grid x:Name="LayoutRoot" Margin="0,0,0,10">
                    <Grid Height="75" VerticalAlignment="Top">
            <i:Interaction.Triggers>
               <i:EventTrigger EventName="Tap">
```

```
                    <Command:EventToCommand Command="{Binding CoursePageDetails}"/>
                </i:EventTrigger>
            </i:Interaction.Triggers>
            <TextBlock x:Name="Title" HorizontalAlignment="Left" Margin="25,-9,0,0"
VerticalAlignment="Top" Height="52" FontSize="37" Text="{Binding Title}"/>
                                <TextBlock x:Name="Professor" HorizontalAlignment="Left"
Margin="25,48,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Text="{Binding
Professor}"/>
                                <Border x:Name="Attendance" Background="{Binding
AttendanceState}" HorizontalAlignment="Left" Height="75" VerticalAlignment="Top"
Width="20"/>
                        </Grid>
            </Grid>
</UserControl>
```

CourseResumeView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CourseResumeView : UserControl
    {
        public CourseResumeView()
        {
            InitializeComponent();
        }
    }
}
```

CourseStudentSituationView.xaml

```
<UserControl x:Class="TesinaMobileCloud.Phone.Client.CourseStudentSituationView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="607" d:DesignWidth="456">

    <Grid x:Name="LayoutRoot">
            <ListBox HorizontalAlignment="Left" Height="607" VerticalAlignment="Top"
Width="456">
```

```
            <TextBlock x:Name="PartialExamResult" Text="Resultado Examen Parcial"
        Style="{StaticResource TextBlockHeader}" Margin="0"/>
            <TextBlock x:Name="PartialExamResultContent" Text="{Binding
        PartialExamResult}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="RecuperationExameResult" Style="{StaticResource
        TextBlockHeader}">
                                <Run Text="Calificación "/>
                                <Run Text="Examen Recup"/>
                                <Run Text="."/>
                        </TextBlock>
            <TextBlock x:Name="RecuperationExameResultContent" Text="{Binding
        RecuperationExamResult}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="FinalExamResult" Text="Resultado Examen Final"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="FinalExamResultContent" Text="{Binding FinalExamResult}"
        Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PracticWorkExameResult" Style="{StaticResource
        TextBlockHeader}">
                                <Run Text="Resultado"/>
                                <Run Text=" "/>
                                <Run Text="de Trab"/>
                                <Run Text="."/>
                                <Run Text=" Practicos"/>
                        </TextBlock>
            <TextBlock x:Name="PracticWorkExameResultContent" Text="{Binding
        PracticWorkExamResult}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="AssistedHoursOfClasses" Text="Total de Horas Asistidas"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="AssistedHoursOfClassesContent" Text="{Binding
        TotalAssistedHours}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="AssistedPercent" Text="Porcentaje de Asistencias"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="AssistedPercentContent" Text="{Binding
        ActualAssistedPercent}" Style="{StaticResource TextBlockContent}"/>
             </ListBox>

       </Grid>
    </UserControl>
```

CourseStudentSituationView.xaml.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Windows;
    using System.Windows.Controls;
    using System.Windows.Navigation;
    using Microsoft.Phone.Controls;
    using Microsoft.Phone.Shell;

    namespace TesinaMobileCloud.Phone.Client
    {
       public partial class CourseStudentSituationView : UserControl
       {
```

```
        public CourseStudentSituationView()
        {
            InitializeComponent();
        }
    }
}
```

CoursesView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    x:Class="TesinaMobileCloud.Phone.Client.View.CoursesView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="498" d:DesignWidth="420"
    DataContext="{Binding Courses, Source={StaticResource Locator}}">
    <Grid x:Name="Courses">
        <toolkit:LongListSelector x:Name="Selector" ItemsSource="{Binding Courses}"
Background="{x:Null}" IsFlatList="True">
            <toolkit:LongListSelector.GroupItemTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding GroupName}" FontSize="32" Foreground="Green"
/>
                </DataTemplate>
            </toolkit:LongListSelector.GroupItemTemplate>
            <toolkit:LongListSelector.ItemTemplate>
                <DataTemplate>
                        <local:CourseResumeView Height="85" Width="432"
toolkit:TiltEffect.IsTiltEnabled="True"/>
                </DataTemplate>
            </toolkit:LongListSelector.ItemTemplate>
        </toolkit:LongListSelector>
    </Grid>
</UserControl>
```

CoursesView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
```

```
using System.Windows.Shapes;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CoursesView : UserControl
    {
        public CoursesView()
        {
            InitializeComponent();
        }
    }
}
```

CourseView.xaml

```
<Client:PhonePage xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                  xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
                  xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
                  xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
                  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                  xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                  xmlns:Client="clr-namespace:TesinaMobileCloud.Phone.Client"
                  xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:ec="clr-
namespace:Microsoft.Expression.Interactivity.Core;assembly=Microsoft.Expression.Interact
ions"
    x:Class="TesinaMobileCloud.Phone.Client.View.CourseView"
    d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    DataContext="{Binding Course, Source={StaticResource Locator}}"
    SupportedOrientations="Portrait"
    Orientation="Portrait"
    mc:Ignorable="d"
    Opacity="0.995"
    Foreground="{StaticResource PhoneControlBackgroundBrush}"
    Background="{StaticResource PhoneControlBackgroundBrush}"
    BorderBrush="{StaticResource PhoneControlBackgroundBrush}"
    shell:SystemTray.IsVisible="True"
          shell:SystemTray.BackgroundColor="{StaticResource ApplicationBarBrush}"
    shell:SystemTray.Opacity="0">
    <toolkit:TransitionService.NavigationInTransition>
        <toolkit:NavigationInTransition>
            <toolkit:NavigationInTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardIn"/>
            </toolkit:NavigationInTransition.Backward>
            <toolkit:NavigationInTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardIn"/>
```

```
            </toolkit:NavigationInTransition.Forward>
        </toolkit:NavigationInTransition>
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
        <toolkit:NavigationOutTransition>
            <toolkit:NavigationOutTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardOut"/>
            </toolkit:NavigationOutTransition.Backward>
            <toolkit:NavigationOutTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardOut"/>
            </toolkit:NavigationOutTransition.Forward>
        </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

    <Client:PhonePage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Mode="Minimized"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="0.35">
            <shell:ApplicationBar.MenuItems>
                    <shell:ApplicationBarMenuItem Text="Inscribirse a examen final"
Click="ApplicationBarIconButton_Click_1" IsEnabled="True"/>
            </shell:ApplicationBar.MenuItems>

                        <shell:ApplicationBarIconButton
IconUri="/Assets/AppBar/appbar.refresh.rest.png" Text="Actualizar"
Click="ApplicationBarIconButton_Click_1"/>
        </shell:ApplicationBar>
    </Client:PhonePage.ApplicationBar>
    <!--LayoutRoot is the root grid where all page content is placed-->
    <shell:SystemTray.ProgressIndicator>
        <shell:ProgressIndicator Text="Actualizando" IsVisible="{Binding Updating}"
IsIndeterminate="True" />
    </shell:SystemTray.ProgressIndicator>
    <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
        <!--Pivot Control-->
        <!--<ProgressBar Background="{StaticResource PhoneControlBackgroundBrush}"
Foreground="#FF033A9B" VerticalAlignment="Top" IsIndeterminate="True"/>-->
        <!--<ProgressBar x:Name="ProgressBar" Background="{StaticResource
PhoneControlBackgroundBrush}" Foreground="#FF033A9B" IsIndeterminate="True"
VerticalAlignment="Top"/>-->
        <controls:Pivot Title="{Binding Title}" SelectedIndex="1" Margin="0,15,0,0">
            <controls:PivotItem Header="Descripción">
                <Client:CourseDescriptionView DataContext="{Binding
CourseDescriptionViewModel}"/>
            </controls:PivotItem>
            <!--Pivot item two-->
            <controls:PivotItem Header="Mi Situación">
                <Client:CourseStudentSituationView DataContext="{Binding
CourseStudentSituationViewModel}"/>
            </controls:PivotItem>
        </controls:Pivot>
    </Grid>
</Client:PhonePage>
```

CourseView.xaml.cs

```
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CourseView : PhonePage
    {
        public CourseView()
        {
            InitializeComponent();
        }

        private void ApplicationBarIconButton_Click_1(object sender, System.EventArgs e)
        {
            var dataContext = DataContext as CourseViewModel;
            if (dataContext != null)
            {
                dataContext.SyncCourseInfo.Execute(null);
            }
        }
    }
}
```

SettingsView.xaml

```
<phone:PhoneApplicationPage xmlns:Primitives="clr-
namespace:Microsoft.Phone.Controls.Primitives;assembly=Microsoft.Phone.Controls.Toolki
t"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    x:Class="TesinaMobileCloud.Phone.Client.SettingsView"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="SIA Mobile" Style="{StaticResource PhoneTextNormalStyle}"/>
```

```
            <TextBlock Text="Configuraciones" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <toolkit:ToggleSwitch HorizontalAlignment="Left" VerticalAlignment="Top"
Margin="0,58,0,0" Width="444"/>
            <TextBlock HorizontalAlignment="Left" Margin="0,10,0,0" TextWrapping="Wrap"
Text="Recibir Notificaciones" VerticalAlignment="Top" FontSize="32"/>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

SettingsView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client
{
    public partial class SettingsView : PhoneApplicationPage
    {
        public SettingsView()
        {
            InitializeComponent();
        }
    }
}
```

ViewModel

CourseDescriptionViewModel.cs

```
using System;
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseDescriptionViewModel : ViewModel
    {
        private string _textToDisplayIfDateTimeIsNotValid = "A Confirmar...";

        /// <summary>
        /// The <see cref="Description" /> property's name.
        /// </summary>
        public const string DescriptionPropertyName = "Description";
```

```
private string _description = string.Empty;

/// <summary>
/// Sets and gets the Description property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Description
{
   get
   {
      return _description;
   }
   set
   {
      Set(() => Description, ref _description, value);
   }
}

/// <summary>
/// The <see cref="Professor" /> property's name.
/// </summary>
public const string ProfessorPropertyName = "Professor";

private string _professor = string.Empty;

/// <summary>
/// Sets and gets the Professor property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Professor
{
   get
   {
      return _professor;
   }
   set
   {
      Set(() => Professor, ref _professor, value);
   }
}

/// <summary>
/// The <see cref="Scheduled" /> property's name.
/// </summary>
public const string ScheduledPropertyName = "Scheduled";

private string _scheduled = string.Empty;

/// <summary>
/// Sets and gets the Scheduled property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Scheduled
```

```
        {
            get
            {
                return _scheduled;
            }

            set
            {
                if (_scheduled == value)
                {
                    return;
                }

                RaisePropertyChanging(() => Scheduled);
                _scheduled = value;
                RaisePropertyChanged(() => Scheduled);
            }
        }

        /// <summary>
        /// The <see cref="PartialExamDate" /> property's name.
        /// </summary>
        public const string PartialExamDatePropertyName = "PartialExamDate";

        private string _partial = string.Empty;

        /// <summary>
        /// Sets and gets the PartialExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string PartialExamDate
        {
            get
            {
                return _partial;
            }
            set
            {
                if (value == DateTime.MinValue.ToShortDateString())
                {
                    Set(() => PracticWorkExamDate, ref _partial,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PracticWorkExamDate, ref _partial, value);
            }
        }

        /// <summary>
        /// The <see cref="RecuperationExamDate" /> property's name.
        /// </summary>
        public const string RecuperationExamDatePropertyName = "RecuperationExamDate";

        private string _recuperationExamDate = string.Empty;
```

```
        /// <summary>
        /// Sets and gets the RecuperationExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string RecuperationExamDate
        {
            get
            {
                return _recuperationExamDate;
            }
            set
            {
                if (value == DateTime.MinValue.ToShortDateString())
                {
                    Set(() => PracticWorkExamDate, ref _recuperationExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PracticWorkExamDate, ref _recuperationExamDate, value);
            }
        }

        /// <summary>
        /// The <see cref="FinalExamDate" /> property's name.
        /// </summary>
        public const string FinalExamDatePropertyName = "FinalExamDate";

        private string _finalExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the FinalExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string FinalExamDate
        {
            get
            {
                return _finalExamDate;
            }
            set
            {
                if (value == DateTime.MinValue.ToShortDateString())
                {
                    Set(() => PracticWorkExamDate, ref _finalExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PracticWorkExamDate, ref _finalExamDate, value);
            }
        }

        /// <summary>
        /// The <see cref="PracticWorkExamDate" /> property's name.
```

```
        /// </summary>
        public const string PracticWorkExamDatePropertyName = "PracticWorkExamDate";

        private string _practicWorkExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the PracticWorkExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string PracticWorkExamDate
        {
            get
            {
                return _practicWorkExamDate;
            }
            set
            {
                if (value == DateTime.MinValue.ToShortDateString())
                {
                    Set(() => PracticWorkExamDate, ref _practicWorkExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PracticWorkExamDate, ref _practicWorkExamDate, value);
            }
        }

        /// <summary>
        /// The <see cref="TotalHoursOfClasses" /> property's name.
        /// </summary>
        public const string TotalHoursOfClassesPropertyName = "TotalHoursOfClasses";

        private int _totalHoursOfClasses;

        /// <summary>
        /// Sets and gets the TotalHoursOfClasses property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public int TotalHoursOfClasses
        {
            get
            {
                return _totalHoursOfClasses;
            }
            set
            {
                Set(() => TotalHoursOfClasses, ref _totalHoursOfClasses, value);
            }
        }

    }
}
```

CourseResumeViewModel.cs

```
using System;
using System.Windows.Media;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseResumeViewModel : ViewModel
    {
        public CourseResumeViewModel()
        {
        }

        /// <summary>
        /// The <see cref="Title" /> property's name.
        /// </summary>
        public const string TitlePropertyName = "Title";

        private string _title = string.Empty;

        /// <summary>
        /// Sets and gets the Title property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Title
        {
            get
            {
                return _title;
            }
            set
            {
                Set(TitlePropertyName, ref _title, value);
            }
        }

        /// <summary>
        /// The <see cref="Professor" /> property's name.
        /// </summary>
        public const string ProfessorPropertyName = "Professor";

        private string _professor = string.Empty;

        /// <summary>
        /// Sets and gets the Professor property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Professor
        {
            get
            {
                return _professor;
```

```
    }
    set
    {
        Set(ProfessorPropertyName, ref _professor, value);
    }
}
/// <summary>
/// The <see cref="Attendance" /> property's name.
/// </summary>
public const string AttendancePropertyName = "Attendance";

private double _attendance = 0;

/// <summary>
/// Sets and gets the Attendance property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public double Attendance
{
    get
    {
        return _attendance;
    }
    set
    {
        Set(AttendancePropertyName, ref _attendance, value);
    }
}

public SolidColorBrush AttendanceState
{
    get
    {
        if (Attendance >= 75)
            return new SolidColorBrush(Colors.Green);
        if (Attendance < 75 && Attendance > 50)
            return new SolidColorBrush(Colors.Yellow);
        return new SolidColorBrush(Colors.Red);
    }
}

/// <summary>
/// The <see cref="Code" /> property's name.
/// </summary>
public const string CodePropertyName = "Code";

private int _code = 0;

/// <summary>
/// Sets and gets the Code property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int Code
{
```

```
    get
    {
        return _code;
    }
    set
    {
        Set(CodePropertyName, ref _code, value);
    }
}

/// <summary>
/// The <see cref="StudentCode" /> property's name.
/// </summary>
public const string StudentCodePropertyName = "StudentCode";

private int _studentCode = 0;

/// <summary>
/// Sets and gets the StudentCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int StudentCode
{
    get
    {
        return _studentCode;
    }
    set
    {
        Set(() => StudentCode, ref _studentCode, value);
    }
}

/// <summary>
/// The <see cref="CareerCode" /> property's name.
/// </summary>
public const string CareerCodePropertyName = "CareerCode";

private int _careerCode = 0;

/// <summary>
/// Sets and gets the CareerCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int CareerCode
{
    get
    {
        return _careerCode;
    }
    set
    {
        Set(() => CareerCode, ref _careerCode, value);
    }
```

```
        }

        private RelayCommand _coursePageDetails;
        public RelayCommand CoursePageDetails
        {
            get
            {
                return _coursePageDetails
                    ?? (_coursePageDetails = new RelayCommand(() =>
NavigationService.NavigateTo(new
Uri(String.Format("/View/CourseView.xaml?CareerCode={0}&StudentCode={1}&CourseCo
de={2}", CareerCode, StudentCode, CareerCode), UriKind.Relative
                    ))));
            }
        }
    }
}
```

CourseStudentSituationViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseStudentSituationViewModel : ViewModel
    {
        /// <summary>
        /// The <see cref="PartialExamResult" /> property's name.
        /// </summary>
        public const string PartialExamResultPropertyName = "PartialExamResult";

        private double _partialExamResult = 0;

        /// <summary>
        /// Sets and gets the PartialExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double PartialExamResult
        {
            get
            {
                return _partialExamResult;
            }
            set
            {
                Set(PartialExamResultPropertyName, ref _partialExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="RecuperationExamResult" /> property's name.
        /// </summary>
        public const string RecuperationExamResultPropertyName =
"RecuperationExamResult";
```

```
        private double _recuperationExamResult = 0;

        /// <summary>
        /// Sets and gets the RecuperationExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double RecuperationExamResult
        {
            get
            {
                return _recuperationExamResult;
            }
            set
            {
                Set(() => RecuperationExamResult, ref _recuperationExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="FinalExamResult" /> property's name.
        /// </summary>
        public const string FinalExamResultPropertyName = "FinalExamResult";

        private double _finalExamResult = 0;

        /// <summary>
        /// Sets and gets the FinalExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double FinalExamResult
        {
            get
            {
                return _finalExamResult;
            }
            set
            {
                Set(() => FinalExamResult, ref _finalExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="PracticWorkExamResult" /> property's name.
        /// </summary>
        public const string PracticWorkExamResultPropertyName =
"PracticWorkExamResult";

        private double _practicWorkExamResult = 0;

        /// <summary>
        /// Sets and gets the PracticWorkExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
```

```
public double PracticWorkExamResult
{
    get
    {
        return _practicWorkExamResult;
    }
    set
    {
        Set(() => PracticWorkExamResult, ref _practicWorkExamResult, value);
    }
}

/// <summary>
/// The <see cref="TotalAssistedHours" /> property's name.
/// </summary>
public const string TotalAssistedHoursPropertyName = "TotalAssistedHours";

private double _totalAssistedHours = 0;

/// <summary>
/// Sets and gets the TotalAssistedHours property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public double TotalAssistedHours
{
    get
    {
        return _totalAssistedHours;
    }
    set
    {
        Set(() => TotalAssistedHours, ref _totalAssistedHours, value);
    }
}

/// <summary>
/// The <see cref="ActualAssistedPercent" /> property's name.
/// </summary>
public const string ActualAssistedPercentPropertyName = "ActualAssistedPercent";

private double _actualAssistedPercent = 0.0;

/// <summary>
/// Sets and gets the ActualAssistedPercent property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public double ActualAssistedPercent
{
    get
    {
        return _actualAssistedPercent;
    }
    set
    {
```

```
                Set(() => ActualAssistedPercent, ref _actualAssistedPercent, value);
            }
        }
    }
}
```

CoursesViewModel.cs

```
using System.Collections.ObjectModel;
using System.Linq;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CoursesViewModel : ViewModel
    {
        public CoursesViewModel(IStudentRepository studentRepository)
        {
            Courses = new ObservableCollection<CourseResumeViewModel>();
            var student = studentRepository.GetStudent();
            var courses = student.Courses;
            if (courses != null)
                courses.ToList().ForEach(c => Courses.Add(new CourseResumeViewModel
                                                {
                                                    Title = c.Course.Title,
                                                    Code = c.Course.Code,
                                                    Professor = c.Course.Professor,
                                                    Attendance = c.Attendance(),
                                                    StudentCode = student.StudentCode,
                                                    CareerCode = student.CareerCode
                                                }));
        }

        public ObservableCollection<CourseResumeViewModel> Courses { get; private set; }
    }
}
```

CourseViewModel.cs

```
using System.Linq;
using System.Windows;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;
using System;
using TesinaMobileCloud.Phone.Services.Support;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseViewModel : ViewModel
    {
        private readonly IStudentCourseRepository _repository;
        private readonly ISyncronizationService _syncService;
```

```
        private readonly IStudentInformationManager _studentInformationManager;

        public CourseViewModel(IStudentCourseRepository repository,
ISyncronizationService syncService, IStudentInformationManager
studentInformationManager)
        {
            _repository = repository;
            _syncService = syncService;
            _studentInformationManager = studentInformationManager;
        }

        /// <summary>
        /// The <see cref="Title" /> property's name.
        /// </summary>
        public const string TitlePropertyName = "Title";
        private string _title = string.Empty;

        /// <summary>
        /// Sets and gets the Title property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Title
        {
            get
            {
                return _title;
            }
            set
            {
                Set(() => Title, ref _title, value);
            }
        }

        /// <summary>
        /// The <see cref="Updating" /> property's name.
        /// </summary>
        public const string UpdatingPropertyName = "Updating";

        private bool _updating = false;

        /// <summary>
        /// Sets and gets the Updating property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public bool Updating
        {
            get
            {
                return _updating;
            }
            set
            {
                Set(UpdatingPropertyName, ref _updating, value);
            }
```

```
        }

        #region Description PivotItem
        public CourseDescriptionViewModel CourseDescriptionViewModel { get; set; }
        #endregion

        #region My Situation PivotItem
        public CourseStudentSituationViewModel CourseStudentSituationViewModel { get;
set; }

        public int Code { get; set; }

        #endregion

        public override void OnNavigateTo(System.Collections.Generic.IDictionary<string,
string> queryString)
        {
            string code;
            if (queryString.TryGetValue("CourseCode", out code))
            {
                var course = _repository.GetCourse(code);
                //var examenParcialTitle = "Examen Parcial";
                Code = int.Parse(code);
                CourseDescriptionViewModel = new CourseDescriptionViewModel
                    {
                        Description = course.Course.Description,
                        FinalExamDate = course.Course.FinalExamDate.ToShortDateString(),
                        PartialExamDate = course.Course.PartialExamDate.ToShortDateString(),
                        PracticWorkExamDate =
course.Course.PracticWorkPresentationDate.ToShortDateString(),
                        Professor = course.Course.Professor,
                        RecuperationExamDate =
course.Course.RecuperationExameDate.ToShortDateString(),
                        Scheduled = course.Course.Scheduled,
                        TotalHoursOfClasses = course.Course.TotalHoursOfClasses
                    };
                CourseStudentSituationViewModel = new CourseStudentSituationViewModel
                    {
                        ActualAssistedPercent = course.Attendance(),
                        FinalExamResult = course.FinalExamResult,
                        PartialExamResult = course.PartialExamResult,
                        RecuperationExamResult = course.RecuperationExamResult,
                        PracticWorkExamResult = course.PracticWorkResult,
                        TotalAssistedHours = course.TotalHoursAssisted

                    };
                Title = course.Course.Title;

                _studentInformationManager.AddCourseExamsDatesReminders(course);
            }

        }

        private RelayCommand _syncCourseInfo;
```

```
        /// <summary>
        /// Gets the MyCommand.
        /// </summary>
        public RelayCommand SyncCourseInfo
        {
            get
            {
                Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; });
                return _syncCourseInfo
                    ?? (_syncCourseInfo = new RelayCommand(() =>
_syncService.SyncCourse(Code).Subscribe(
                        (StudentCourse x) =>
                        {
                            //DescriptionViewModel
                            CourseDescriptionViewModel.Description = x.Course.Description;
                            CourseDescriptionViewModel.FinalExamDate =
x.Course.FinalExamDate.ToShortDateString();
                            CourseDescriptionViewModel.PartialExamDate =
x.Course.PartialExamDate.ToShortDateString();
                            CourseDescriptionViewModel.Professor = x.Course.Professor;
                            CourseDescriptionViewModel.PracticWorkExamDate =
x.Course.PracticWorkPresentationDate.ToShortDateString();
                            CourseDescriptionViewModel.Scheduled = x.Course.Scheduled;
                            CourseDescriptionViewModel.RecuperationExamDate =
x.Course.RecuperationExameDate.ToShortDateString();

                            //MySituationViewModel
                            CourseStudentSituationViewModel.ActualAssistedPercent =
x.Attendance();
                            CourseStudentSituationViewModel.FinalExamResult =
x.FinalExamResult;
                            CourseStudentSituationViewModel.PartialExamResult =
x.PartialExamResult;
                            CourseStudentSituationViewModel.PracticWorkExamResult =
x.PracticWorkResult;
                            CourseStudentSituationViewModel.RecuperationExamResult =
x.RecuperationExamResult;
                            CourseStudentSituationViewModel.TotalAssistedHours =
x.TotalHoursAssisted;

                            _studentInformationManager.AddCourseExamsDatesReminders(x);

                        },
                        (Exception ex) => Deployment.Current.Dispatcher.BeginInvoke(() => {
Updating = false; }),
                        () => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false;
})));
            }
        }


    }
}
```

MainViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class MainViewModel : ViewModel
    {
        public MainViewModel()
        {
        }
    }
}
```

ViewModel.cs

```
using System.Collections.Generic;
using GalaSoft.MvvmLight;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;
using GalaSoft.MvvmLight.Ioc;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public abstract class ViewModel : ViewModelBase
    {
        protected readonly INavigationService NavigationService;

        protected ViewModel()
        {
            NavigationService = SimpleIoc.Default.GetInstance<INavigationService>();
        }

        public virtual void OnNavigateTo(IDictionary<string, string> queryString)
        {
        }

        public virtual void OnNavigateFrom(IDictionary<string, string> queryString)
        {
        }
    }
}
```

ViewModelLocator.cs

```
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;
using TesinaMobileCloud.Phone.Client.ViewServices;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;
using TesinaMobileCloud.Phone.Data.Repositories;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
```

```
{
   public class ViewModelLocator
   {
      static ViewModelLocator()
      {
         ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

         // Create run time view services and models
         SimpleIoc.Default.Register<IStudentCourseRepository,
MockStudentCourseRepository>();
         SimpleIoc.Default.Register<IStudentRepository, MockStudentRepository>();
         SimpleIoc.Default.Register<IScheduleActionClient, ScheduleActionClient>();
         SimpleIoc.Default.Register<IScheduleActionService,
ScheduleActionServiceAdapter>();
         SimpleIoc.Default.Register<INavigationService, NavigationService>();
         SimpleIoc.Default.Register<ISyncronizationService, SyncronizationService>();
         SimpleIoc.Default.Register<ICloudService, CloudService>();
         SimpleIoc.Default.Register<IStudentInformationManager,
StudentInformationManager>();


         SimpleIoc.Default.Register<MainViewModel>();
         SimpleIoc.Default.Register<CoursesViewModel>();
         SimpleIoc.Default.Register<CourseResumeViewModel>();
         SimpleIoc.Default.Register<CourseViewModel>();
         SimpleIoc.Default.Register<CourseStudentSituationViewModel>();
         SimpleIoc.Default.Register<CourseDescriptionViewModel>();
      }

      public MainViewModel Main
      {
         get
         {
            return ServiceLocator.Current.GetInstance<MainViewModel>();
         }
      }
      public CoursesViewModel Courses
      {
         get
         {
            return ServiceLocator.Current.GetInstance<CoursesViewModel>();
         }
      }
      public CourseResumeViewModel CourseResume
      {
         get
         {
            return ServiceLocator.Current.GetInstance<CourseResumeViewModel>();
         }

      }
      public CourseViewModel Course
      {
         get
```

```
            {
                return ServiceLocator.Current.GetInstance<CourseViewModel>();
            }

        }
        public CourseDescriptionViewModel CourseDescription
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CourseDescriptionViewModel>();
            }

        }
        public CourseStudentSituationViewModel CourseStudentSituation
        {
            get
            {
                return
    ServiceLocator.Current.GetInstance<CourseStudentSituationViewModel>();
            }

        }
        public IScheduleActionClient ScheduleActions
        {
            get
            {
                return ServiceLocator.Current.GetInstance<IScheduleActionClient>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

ViewServices

NavigationService.cs

```
using Microsoft.Phone.Controls;
using System;
using System.Windows;
using System.Windows.Navigation;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewServices
{
    public class NavigationService : INavigationService
    {
        private PhoneApplicationFrame _mainFrame;

        public event NavigatingCancelEventHandler Navigating;
```

```
        public void NavigateTo(Uri pageUri)
        {
            if (InitializatedFrame())
            {
                _mainFrame.Navigate(pageUri);
            }
        }

        public void GoBack()
        {
            if (InitializatedFrame() && _mainFrame.CanGoBack)
            {
                _mainFrame.GoBack();
            }
        }

        private bool InitializatedFrame()
        {
            if (_mainFrame != null)
                return true;

            _mainFrame = Application.Current.RootVisual as PhoneApplicationFrame;

            if (_mainFrame != null)
            {
                _mainFrame.Navigating += (s, e) =>
                {
                    if (Navigating != null)
                    {
                        Navigating(s, e);
                    }
                };

                return true;
            }

            return false;
        }
    }
}
```

INavigationService.cs

```
using System;
using System.Windows.Navigation;

namespace TesinaMobileCloud.Phone.Client.ViewServices.Interfaces
{
    public interface INavigationService
    {
        event NavigatingCancelEventHandler Navigating;
        void NavigateTo(Uri pageUri);
        void GoBack();
    }
```

```
}
```

TesinaMobileCloud.Phone.Client.Test

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.Test.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test.ViewModel" mc:Ignorable="d">
  <!--Application Resources-->
  <Application.Resources>
    <ResourceDictionary>
      <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
      <ResourceDictionary.MergedDictionaries></ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
  <Application.ApplicationLifetimeObjects>
    <!--Required object that handles lifetime events for the application-->
    <shell:PhoneApplicationService Launching="Application_Launching"
Closing="Application_Closing" Activated="Application_Activated"
Deactivated="Application_Deactivated" />
  </Application.ApplicationLifetimeObjects>
</Application>
```

App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }
```

```
        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
                // application's PhoneApplicationService object to Disabled.
                // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
                // and consume battery power when the user is not using the phone.
                PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
            }

        }

        // Code to execute when the application is launching (eg, from Start)
        // This code will not execute when the application is reactivated
        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
        }

        // Code to execute when the application is activated (brought to foreground)
        // This code will not execute when the application is first launched
        private void Application_Activated(object sender, ActivatedEventArgs e)
        {
        }

        // Code to execute when the application is deactivated (sent to background)
```

```
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEventArgs e)
        {
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
        {
        }

        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // A navigation has failed; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
```

```
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
    e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="TesinaMobileCloud.Phone.Client.Test.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Testing;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            this.Content = UnitTestSystem.CreateTestPage();
        }
    }
}
```

CoursesViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.ObjectModel;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CoursesViewModelFixture
    {
        readonly IStudentRepository _studentRepository = new MockStudentRepository();

        [TestMethod]
        public void CoursesViewModel()
        {
            var sut = new CoursesViewModel(_studentRepository);

            Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
            Assert.IsInstanceOfType(sut.Courses,
typeof(ObservableCollection<CourseResumeViewModel>));
        }

        [TestMethod]
        public void CoursesViewModelFillCoursesList()
```

```
        {
            var sut = new CoursesViewModel(_studentRepository);

            Assert.IsNotNull(sut.Courses);
            Assert.AreEqual(1, sut.Courses.Count);
        }
    }
}
```

CourseViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Windows.Media;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CourseViewModelFixture
    {

        [TestMethod]
        public void CourseViewModelWithParams()
        {
            var sut = new CourseResumeViewModel
            {
                Code = 124,
                Title = "Title",
                Professor = "Prof",
                //YearOfCareer = 1,
                Attendance = 1
            };

            Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
            Assert.IsNotNull(sut);
            Assert.AreEqual(124, sut.Code);
            Assert.AreEqual("Title", sut.Title);
            //Assert.AreEqual(1, sut.YearOfCareer);
            Assert.AreEqual("Prof", sut.Professor);
            Assert.AreEqual(1, sut.Attendance);
            Assert.IsInstanceOfType(sut.AttendanceState, typeof(SolidColorBrush));
        }

        [TestMethod]
        public void CourseViewModelWithOutParams()
        {
            var sut = new CourseResumeViewModel();
            Assert.IsNotNull(sut);
        }
    }
}
```

MainViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
    /// <summary>
    /// This class contains properties that the main View can data bind to.
    /// <para>
    /// Use the <strong>mvvminpc</strong> snippet to add bindable properties to this
ViewModel.
    /// </para>
    /// <para>
    /// You can also use Blend to data bind with the tool's support.
    /// </para>
    /// <para>
    /// See http://www.galasoft.ch/mvvm
    /// </para>
    /// </summary>
    public class MainViewModel : ViewModelBase
    {
        /// <summary>
        /// Initializes a new instance of the MainViewModel class.
        /// </summary>
        public MainViewModel()
        {
            ////if (IsInDesignMode)
            ////{
            ////    // Code runs in Blend --> create design time data.
            ////}
            ////else
            ////{
            ////    // Code runs "for real"
            ////}
        }
    }
}
```

ViewModelLocator.cs

```
/*
  In App.xaml:
  <Application.Resources>
      <vm:ViewModelLocator xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test"
                           x:Key="Locator" />
  </Application.Resources>

  In the View:
  DataContext="{Binding Source={StaticResource Locator}, Path=ViewModelName}"

  You can also use Blend to do all this with the tool's support.
  See http://www.galasoft.ch/mvvm
*/

using GalaSoft.MvvmLight;
```

```
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;

namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
    /// <summary>
    /// This class contains static references to all the view models in the
    /// application and provides an entry point for the bindings.
    /// </summary>
    public class ViewModelLocator
    {
        /// <summary>
        /// Initializes a new instance of the ViewModelLocator class.
        /// </summary>
        public ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            ////if (ViewModelBase.IsInDesignModeStatic)
            ////{
            ////    // Create design time view services and models
            ////    SimpleIoc.Default.Register<IDataService, DesignDataService>();
            ////}
            ////else
            ////{
            ////    // Create run time view services and models
            ////    SimpleIoc.Default.Register<IDataService, DataService>();
            ////}

            SimpleIoc.Default.Register<MainViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

TesinaMobileCloud.Phone.Data

MockStudentCourseRepository.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;
```

```
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
   public class MockStudentCourseRepository : IStudentCourseRepository
   {
      private static List<StudentCourse> _courses;

      public MockStudentCourseRepository()
      {
         _courses = new List<StudentCourse>
         {
            new StudentCourse
            {
               Course = new Course
               {
                  Code = 1,
                  Title = "Programación I",
                  Professor = "Ing. Carlos Rodrigo",
                  Description = "Programacion I introduce los conceptos basicos de la
programacion, "+
                  "comenzado con el paradigma estructurado y luego introduciendo el
paradigma de objetos. "+
                  "El lenguaje de programacion a utilizar es Java.",
                  TotalHoursOfClasses = 60,
                  Scheduled = "Miercoles 17Hs",
                  FinalExamDate =new DateTime(2013,02,4,00,22,00),
                  PartialExamDate = new DateTime(2013,02,4,00,21,00),
                  PracticWorkPresentationDate = DateTime.MinValue,
                  RecuperationExameDate = DateTime.MinValue
               },
               FinalExamResult = 10,
               PartialExamResult = 9,
               PracticWorkResult = 9,
               RecuperationExamResult = 0,
               TotalHoursAssisted = 40
            }
         };
      }

      public IEnumerable<StudentCourse> GetCourses()
      {
         return _courses;
      }

      public void AddCourses(IEnumerable<StudentCourse> courses)
      {
         _courses.Clear();
         foreach (var course in courses)
            _courses.Add(course);
      }

      public void AddCourse(StudentCourse course)
      {
```

```
            throw new NotImplementedException();
        }

        public StudentCourse GetCourse(string code)
        {
            return new StudentCourse
                {
                    Course = new Course
                        {
                            Code = 1,
                            Title = "Programación I",
                            Professor = "Ing. Carlos Rodrigo",
                            Description = "Programacion I introduce los conceptos basicos de la
programacion, " +
                                          "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. " +
                                          "El lenguaje de programacion a utilizar es Java.",
                            TotalHoursOfClasses = 60,
                            Scheduled = "Miercoles 17Hs",
                            FinalExamDate = new DateTime(2013, 03, 6, 00, 28, 00),
                            PartialExamDate = new DateTime(2013, 03, 6, 00, 26, 00),
                            PracticWorkPresentationDate = DateTime.MinValue,
                            RecuperationExameDate = DateTime.MinValue
                        },
                    FinalExamResult = 10,
                    PartialExamResult = 9,
                    PracticWorkResult = 9,
                    RecuperationExamResult = 0,
                    TotalHoursAssisted = 40
                };
        }
    }

}
```

StudentCourseRepository.cs

```
using System.Collections.Generic;
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class StudentCourseRepository : IStudentCourseRepository
    {
        private const string studentcourseFormat = "StudentCourse{0}";
        private readonly IsolatedStorageSettings _isolatedStorage;

        public StudentCourseRepository()
        {
            _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
        }
```

```
        public IEnumerable<StudentCourse> GetCourses()
        {
            IEnumerable<StudentCourse> courses;
            _isolatedStorage.TryGetValue("StudentCourse", out courses);
            return courses;
        }

        public void AddCourses(IEnumerable<StudentCourse> courses)
        {
            foreach (var course in courses)
            {
                AddCourseToIsolatedStorage(course);
            }
            _isolatedStorage.Save();
        }

        public void AddCourse(StudentCourse course)
        {
            AddCourseToIsolatedStorage(course);
            _isolatedStorage.Save();
        }

        public StudentCourse GetCourse(string code)
        {
            StudentCourse course;
            var courseCode = string.Format(studentcourseFormat, code);
            _isolatedStorage.TryGetValue(courseCode, out course);
            return course;
        }

        private void AddCourseToIsolatedStorage(StudentCourse course)
        {
            StudentCourse studentCourse;
            var courseCode = string.Format(studentcourseFormat, course.Course.Code);
            if (_isolatedStorage.TryGetValue(courseCode, out studentCourse))
            {
                _isolatedStorage.Remove(courseCode);
            }
            _isolatedStorage.Add(courseCode, course);
        }
    }
}
```

StudentRepository.cs

```
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class StudentRepository : IStudentRepository
    {
        private readonly IsolatedStorageSettings _isolatedStorage;
```

```
        public StudentRepository()
        {
            _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
        }

        public void AddStudent(Student student)
        {
            Student studentToSave;
            if (_isolatedStorage.TryGetValue("Student", out studentToSave))
            {
                _isolatedStorage.Remove("Student");
            }
            _isolatedStorage.Add("Student", student);
            _isolatedStorage.Save();
        }

        public Student GetStudent()
        {
            Student student;
            _isolatedStorage.TryGetValue("Student", out student);
            return student;
        }
    }
}
```

Interfaces

IStudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Repositories.Interfaces
{
    public interface IStudentCourseRepository
    {
        void AddCourses(IEnumerable<StudentCourse> courses);
        void AddCourse(StudentCourse course);
        StudentCourse GetCourse(string code);
    }
}
```

IStudentRepository.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Repositories.Interfaces
{
    public interface IStudentRepository
    {
        void AddStudent(Student student);
        Student GetStudent();
    }
```

```
public class MockStudentRepository : IStudentRepository
{
    public void AddStudent(Student student)
    {
        throw new System.NotImplementedException();
    }

    public Student GetStudent()
    {
        return new Student
            {
                Courses = new List<StudentCourse>
                        {
                            new StudentCourse
                            {
                                Course = new Course
                                {
                                    Code = 1,
                                    Title = "Programación I",
                                    Professor = "Ing. Carlos Rodrigo",
                                    Description = "Programacion I introduce los conceptos basicos
de la programacion, "+
                                    "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. "+
                                    "El lenguaje de programacion a utilizar es Java.",
                                    TotalHoursOfClasses = 60,
                                    Scheduled = "Miercoles 17Hs",
                                    FinalExamDate = new DateTime(2013,02,4,00,20,00),
                                    PartialExamDate = new DateTime(2013,02,4,00,21,00),
                                    PracticWorkPresentationDate = DateTime.MinValue,
                                    RecuperationExameDate = DateTime.MinValue
                                },
                                FinalExamResult = 10,
                                PartialExamResult = 9,
                                PracticWorkResult = 9,
                                RecuperationExamResult = 0,
                                TotalHoursAssisted = 40
                            }
                        },
                CareerCode = 10004,
                StudentCode = 502,
                FirstName = "Carlos Sergio",
                LastName = "Rodrigo"
            };
    }
}
}
```

TesinaMobileCloud.Phone.Data.Test

UnitTest1.cs

```
using System.Collections.Generic;
```

```
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Test
{
   [TestClass]
   public class StudentRepositoryFixture
   {
      [TestMethod]
      public void GetCourses()
      {
         var sut = new Data.Repositories.MockStudentCourseRepository();

         var result = sut.GetCourses();

         Assert.IsNotNull(result);
         Assert.IsInstanceOfType(result, typeof(IEnumerable<StudentCourse>));
      }

      [TestMethod]
      public void AddCourses()
      {
         var courses = new List<StudentCourse>
                     {
                        new StudentCourse
                        {
                           Course = new Course{
                           Title = "Programacion I", Professor = "Prof. Diana Ciccinelli"}
                        },
                        new StudentCourse
                        {
                           Course = new Course{
                           Title = "Programacion II", Professor = "Prof. Martin Cernadas"}
                        }
                     };
         var sut = new Data.Repositories.MockStudentCourseRepository();

         sut.AddCourses(courses);

         var result = sut.GetCourses();
         Assert.IsNotNull(result);
         Assert.AreEqual(2, result.Count());
      }
   }
}
```

TesinaMobileCloud.Phone.Services

CloudService.cs

```
using System;
using System.Net;
using System.Runtime.Serialization.Json;
```

```
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
   public class CloudService : ICloudService
   {
      public IObservable<Student> GetAllCourses(Uri baseUri)
      {
         var relativeUri = String.Format("RetriveStudentByCodeAndCareer/{0}/{1}", "10004",
"502");//TODO: Alto HardCoding, que lo tome del IsolatedStorage

         var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
         request.Method = "GET";
         request.Accept = "application/json";

         return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
            .Select(
               response =>
               {
                  using (var responseStream = response.GetResponseStream())
                  {
                     var serializer = new DataContractJsonSerializer(typeof(Student));
                     return serializer.ReadObject(responseStream) as Student;
                  }
               }
            );
      }

      public IObservable<StudentCourse> GetCourse(Uri baseUri, int code)
      {
         var relativeUri = String.Format("RetriveStudentCourse/{0}", code.ToString());
         var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
         request.Method = "GET";
         request.Accept = "application/json";

         return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
            .Select(
               response =>
               {
                  using (var responseStream = response.GetResponseStream())
                  {
                     var serializer = new DataContractJsonSerializer(typeof(StudentCourse));
                     return serializer.ReadObject(responseStream) as StudentCourse;
                  }
               }
            );
      }
   }
}
```

ScheduleActionClient.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionClient : IScheduleActionClient
    {
        private const string PeriodicTaskName = "SyncCourses";
        private readonly IScheduleActionService _scheduleActionService;

        public ScheduleActionClient(IScheduleActionService scheduleActionService)
        {
            _scheduleActionService = scheduleActionService;
        }

        public void AddPeriodicTask()
        {
            var periodicTask = new PeriodicTask(PeriodicTaskName)
                               {
                                   Description = "Syncronization Courses information Task"
                               };
            if (_scheduleActionService.Find(PeriodicTaskName) != null)
                _scheduleActionService.Remove(PeriodicTaskName);

            _scheduleActionService.Add(periodicTask);

#if DEBUG
            _scheduleActionService.LaunchForTest(PeriodicTaskName,
TimeSpan.FromMinutes(1));
#endif
        }

        public void RemovePeriodicTask()
        {
            if (_scheduleActionService.Find(PeriodicTaskName) != null)
                _scheduleActionService.Remove(PeriodicTaskName);
        }
    }
}
```

ScheduleActionServiceAdapter.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionServiceAdapter : IScheduleActionService
    {
        public ScheduledAction Find(string taskName)
        {
            return ScheduledActionService.Find(taskName);
```

```
        }

        public void Add(ScheduledAction action)
        {
            ScheduledActionService.Add(action);
        }

        public void Remove(string taskName)
        {
            ScheduledActionService.Remove(taskName);
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            ScheduledActionService.LaunchForTest(actionName, delay);
        }
    }
}
```

StudentInformationManager.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class StudentInformationManager : IStudentInformationManager
    {
        private readonly IStudentRepository _studentRepository;
        private readonly IStudentCourseRepository _studentCourseRepository;
        private readonly IScheduleActionService _scheduleActionService;

        public StudentInformationManager(IStudentRepository studentRepository,
IStudentCourseRepository studentCourseRepository, IScheduleActionService
scheduleActionService)
        {
            _studentRepository = studentRepository;
            _studentCourseRepository = studentCourseRepository;
            _scheduleActionService = scheduleActionService;
        }

        public string ProcessStudentInformation(Student student)
        {
            _studentRepository.AddStudent(student);
            foreach (var studentCourse in student.Courses)
            {
                ProcessStudentCourseInformation(studentCourse);
            }

            return string.Empty;
        }
```

```
        public void ProcessStudentCourseInformation(StudentCourse studentCourse)
        {
            _studentCourseRepository.AddCourse(studentCourse);
        }

        public void AddCourseExamsDatesReminders(StudentCourse studentCourse)
        {
            var courseTitle = studentCourse.Course.Title;
            if (studentCourse.Course.PartialExamDate != DateTime.MinValue)
                AddRemainderForCourseExam(String.Format("{0}ExamParcial",
studentCourse.Course.Code),
                                          String.Format("Examen Parcial de {0}", courseTitle),
                                          String.Format("En d�a de ma�ana se dicta el Examen Parcial
de {0}", courseTitle),
                                          studentCourse.Course.PartialExamDate,
                                          24);
            if (studentCourse.Course.RecuperationExameDate != DateTime.MinValue)
                AddRemainderForCourseExam(String.Format("{0}ExamRecuperatorio",
studentCourse.Course.Code),
                                          String.Format("Examen Recuperatorio de {0}", courseTitle),
                                          String.Format("En d�a de ma�ana se dicta el Examen
Recuperatorio de {0}", courseTitle),
                                          studentCourse.Course.RecuperationExameDate,
                                          24);
            if (studentCourse.Course.PracticWorkPresentationDate != DateTime.MinValue)

AddRemainderForCourseExam(String.Format("{0}PresentacionTrabajosPracticos",
studentCourse.Course.Code),
                                          String.Format("Presentaci�n de Trabajos Practicos de {0}",
courseTitle),
                                          String.Format("En d�a de ma�ana se realiza la Presentaci�n
de Trabajos Practicos de {0}", courseTitle),
                                          studentCourse.Course.PracticWorkPresentationDate,
                                          24);

            if (studentCourse.Course.FinalExamDate != DateTime.MinValue)
            {
                if (!studentCourse.FinalExamInscripted)
                {
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal1",
studentCourse.Course.Code),
                                              String.Format("Examen Final de {0}", courseTitle),
                                              String.Format("Recuerde inscribirse al Examen Final de {0}",
courseTitle),
                                              studentCourse.Course.FinalExamDate,
                                              96);
                AddRemainderForCourseExam(String.Format("{0}ExamFinal2",
studentCourse.Course.Code),
                                          String.Format("Examen Final de {0}", courseTitle),
                                          String.Format("Recuerde inscribirse al Examen Final de {0}",
courseTitle),
                                          studentCourse.Course.FinalExamDate,
                                          84);
```

```
                AddRemainderForCourseExam(String.Format("{0}ExamFinal3",
studentCourse.Course.Code),
                                          String.Format("Examen Final de {0}", courseTitle),
                                          String.Format("Recuerde inscribirse al Examen Final de {0}",
courseTitle),
                                          studentCourse.Course.FinalExamDate,
                                          73);
            }
            else
            {
                _scheduleActionService.Remove(String.Format("{0}ExamFinal1", courseTitle));
                _scheduleActionService.Remove(String.Format("{0}ExamFinal2", courseTitle));
                _scheduleActionService.Remove(String.Format("{0}ExamFinal3", courseTitle));
            }

            AddRemainderForCourseExam(String.Format("{0}ExamFinal",
studentCourse.Course.Code),
                                      String.Format("Examen Final de {0}", courseTitle),
                                      String.Format("En d�a de ma�ana se dicta el Examen Final de
{0}", courseTitle),
                                      studentCourse.Course.FinalExamDate,
                                      24);
        }
    }

    private void AddRemainderForCourseExam(string remainderName, string title, string
content, DateTime examDate, int hoursEarlyToRemaind)
    {
        var firstPartialReminder = remainderName;

        if (_scheduleActionService.Find(firstPartialReminder) != null)
            _scheduleActionService.Remove(firstPartialReminder);
        _scheduleActionService.Add(new Reminder(firstPartialReminder)
            {
                BeginTime = examDate.AddHours(-hoursEarlyToRemaind),
                Title = title,
                Content = content
            });
    }
  }
}
```

SyncronizationService.cs

```
using System;
using System.Linq;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services
{
  public class SyncronizationService : ISyncronizationService
```

```
    {
        private readonly ICloudService _cloudService;
        private readonly IStudentInformationManager _studentInformationManager;
        private readonly Uri _baseUri = new Uri("http://168.62.53.122:8081/Service1.svc/");

        public SyncronizationService(ICloudService cloudService,
IStudentInformationManager studentInformationManager)
        {
            _cloudService = cloudService;
            _studentInformationManager = studentInformationManager;
        }

        public IObservable<TaskSummary> SyncCourses()
        {
            var resuls = _cloudService.GetAllCourses(_baseUri)
                .Select(student =>
                    {
                        var message =
_studentInformationManager.ProcessStudentInformation(student);
                        return new TaskSummary
                                {
                                    Result = TaskResult.Success
                                };
                    })
                .Catch((Exception exception) =>
                        {
                            throw exception;
                        });

            return resuls;
        }

        public IObservable<StudentCourse> SyncCourse(int code)
        {
            var result = _cloudService.GetCourse(_baseUri, code)
                                    .Select(course =>
                                        {

_studentInformationManager.ProcessStudentCourseInformation(course);
                                            return course;
                                        })
                                    .Catch((Exception exception) =>
                                    {
                                        throw exception;
                                    });
            return result;
        }
    }
}
```

Interfaces

ICloudService.cs

```
using System;
```

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ICloudService
    {
        IObservable<Student> GetAllCourses(Uri baseUri);
        IObservable<StudentCourse> GetCourse(Uri baseUri, int code);
    }
}
```

IScheduleActionClient.cs

```
namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionClient
    {
        void AddPeriodicTask();
        void RemovePeriodicTask();
    }
}
```

IScheduleActionService.cs

```
using System;
using Microsoft.Phone.Scheduler;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionService
    {
        ScheduledAction Find(string taskName);
        void Add(ScheduledAction action);
        void Remove(string taskName);
        void LaunchForTest(string actionName, TimeSpan delay);
    }
}
```

IStudentInformationManager.cs

```
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IStudentInformationManager
    {
        string ProcessStudentInformation(Student student);
        void ProcessStudentCourseInformation(StudentCourse studentCourse);
        void AddCourseExamsDatesReminders(StudentCourse studentCourse);
    }
}
```

ISyncronizationService.cs

```
using System;
using TesinaMobileCloud.Data.DTO;
```

```
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ISyncronizationService
    {
        IObservable<TaskSummary> SyncCourses();
        IObservable<StudentCourse> SyncCourse(int code);
    }
}
```

Mocks

MockCloudService.cs

```
using Microsoft.Phone.Reactive;
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockCloudService : ICloudService
    {
        public IObservable<Student> GetAllCourses(Uri baseUri)
        {
            return Observable.ToObservable(new List<Student>{new Student
                {
                    Courses = new List<StudentCourse>
                        {
                            new StudentCourse
                            {
                                Course = new Course
                                {
                                    Code = 1,
                                    Title = "Programaci�n I",
                                    Professor = "Ing. Carlos Rodrigo",
                                    Description = "Programacion I introduce los conceptos basicos
de la programacion, "+
                                    "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. "+
                                    "El lenguaje de programacion a utilizar es Java.",
                                    TotalHoursOfClasses = 60,
                                    Scheduled = "Miercoles 17Hs",
                                    FinalExamDate = new DateTime(2013,02,04,00,05,00),
                                    PartialExamDate = new DateTime(2013,02,04,00,08,00),
                                    PracticWorkPresentationDate = new
DateTime(2013,02,04,00,07,00),
                                    RecuperationExameDate = new DateTime(2013,02,04,00,06,00)
                                },
                                FinalExamResult = 10,
                                PartialExamResult = 9,
                                PracticWorkResult = 9,
                                RecuperationExamResult = 0,
```

```
                                TotalHoursAssisted = 40
                            }
                        }
                    }));
            }

            public IObservable<StudentCourse> GetCourse(Uri baseUri, int code)
            {
                throw new NotImplementedException();
            }
        }
    }
```

MockScheduleActionService.cs

```
    using System;
    using Microsoft.Phone.Scheduler;
    using TesinaMobileCloud.Phone.Services.Interfaces;

    namespace TesinaMobileCloud.Phone.Services.Mocks
    {
        public class MockScheduleActionService : IScheduleActionService
        {
            private PeriodicTask periodic;
            public ScheduledAction Find(string taskName)
            {
                return periodic;
            }

            public void Add(ScheduledAction action)
            {
                periodic = new PeriodicTask("MockAdd");
            }

            public void Remove(string taskName)
            {
                periodic = null;
            }

            public void LaunchForTest(string actionName, TimeSpan delay)
            {
                //throw new NotImplementedException();
            }
        }
    }
```

MockSyncronizationService.cs

```
    using System;
    using Microsoft.Phone.Reactive;
    using TesinaMobileCloud.Phone.Services.Interfaces;
    using TesinaMobileCloud.Phone.Services.Support;
    using TesinaMobileCloud.Data.DTO;

    namespace TesinaMobileCloud.Phone.Services.Mocks
    {
```

```
    public class MockSyncronizationService : ISyncronizationService
    {
        public IObservable<TaskSummary> SyncCourses()
        {
            return Observable.Return(new TaskSummary
                                     {
                                        Result = TaskResult.Success
                                     });
        }

        public IObservable<StudentCourse> SyncCourse(int code)
        {
            throw new NotImplementedException();
        }
    }
}
```

Support

TaskResult.cs

```
namespace TesinaMobileCloud.Phone.Services.Support
{
    public enum TaskResult
    {
        Success,
        Failed
    }
}
```

TaskSummary.cs

```
namespace TesinaMobileCloud.Phone.Services.Support
{
    public class TaskSummary
    {
        public TaskResult Result { get; set; }
    }
}
```

TesinaMobileCloud.Phone.Services.Test

CloudServiceFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class CloudServiceFixture
    {
        [TestMethod]
        public void GetAllCourses()
```

```
        {
            var sut = new MockCloudService();
            var result = sut.GetAllCourses(new Uri("http://127.0.0.1"));
            Assert.IsNotNull(result);
        }
    }
}
```

ScheduleActionsServiceFixture.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class ScheduleActionsServiceFixture
    {
        public Interfaces.IScheduleActionService mockScheduleService = new
MockScheduleActionService();

        [TestMethod]
        public void RemoveAgent()
        {
            var sut = new ScheduleActionClient(mockScheduleService);

            sut.RemovePeriodicTask();

            Assert.IsNull(mockScheduleService.Find("MockRemove"));

        }

        [TestMethod]
        public void AddAgent()
        {
            var sut = new ScheduleActionClient(mockScheduleService);

            sut.AddPeriodicTask();

            Assert.IsNotNull(mockScheduleService.Find("MockAdd"));
        }

    }
}
```

SyncronizationServiceFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Test
{
```

```
    [TestClass]
    public class SyncronizationServiceFixture
    {
        [TestMethod]
        public void SyncCourses()
        {
            var sut = new MockSyncronizationService();

            var result = sut.SyncCourses();

            Assert.IsNotNull(result);
        }

        //[TestMethod]
        //public void SyncCourse()
        //{
        //    var code = 1;
        //    var sut = new MockSyncronizationService();

        //    var result = sut.SyncCourse(code);

        //    Assert.IsNotNull(result);
        //}
    }
}
```

TesinaMobileCloud.Phone.Shared

Class1.cs

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using GalaSoft.MvvmLight.Ioc;

namespace TesinaMobileCloud.Phone.Shared
{
    public class CustomIocContainer : SimpleIoc
    {
        static CustomIocContainer()
        {

        }
    }
}
```

TesinaMobileCloud.PushNotificationSenderRole

WorkerRole.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;
using Microsoft.WindowsAzure.StorageClient;
using Ninject;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Shared;

namespace TesinaMobileCloud.PushNotificationSenderRole
{
    public class WorkerRole : RoleEntryPoint
    {
        private IKernel _container;

        public override void Run()
        {
            // This is a sample worker implementation. Replace with your logic.
            Trace.WriteLine("TesinaMobileCloud.PushNotificationSenderRole entry point
called", "Information");
            var queue = _container.Get<IQueue<QueueNotificationMessage>>();
            while (true)
            {
                Thread.Sleep(10000);
                Trace.WriteLine("Working", "Information");
                if (!queue.HasMessage) continue;
                var notificationMessage = queue.GetMessage();

                try
                {
                    var sendNotificationRequest =
(HttpWebRequest)WebRequest.Create(notificationMessage.Destinatary);
                    byte[] notificationMessageInBytes =
Encoding.Default.GetBytes(notificationMessage.Payload);

                    // Set the web request content length.
                    sendNotificationRequest.ContentLength = notificationMessageInBytes.Length;
                    sendNotificationRequest.ContentType = "text/xml";
                    sendNotificationRequest.Headers.Add("X-WindowsPhone-Target",
notificationMessage.Type);
                    sendNotificationRequest.Headers.Add("X-NotificationClass", "1");
```

```
                using (var requestStream = sendNotificationRequest.GetRequestStream())
                {
                    requestStream.Write(notificationMessageInBytes, 0,
notificationMessageInBytes.Length);
                }

                // Send the notification and get the response.
                var response = (HttpWebResponse)sendNotificationRequest.GetResponse();
                var notificationStatus = response.Headers["X-NotificationStatus"];
                //var notificationChannelStatus = response.Headers["X-SubscriptionStatus"];
                var deviceConnectionStatus = response.Headers["X-
DeviceConnectionStatus"];
                if(!notificationStatus.Equals("Connected") ||
!deviceConnectionStatus.Equals("Received"))
                    queue.AddMessage(notificationMessage);
            }
            catch (Exception)
            {
                queue.AddMessage(notificationMessage);
            }
        }
    }

    public override bool OnStart()
    {
        // Set the maximum number of concurrent connections
        ServicePointManager.DefaultConnectionLimit = 12;

        // For information on handling configuration changes
        // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.
        _container = new StandardKernel(new ServiceModule());
        return base.OnStart();
    }
  }
}
```

TesinaMobileCloud.Shared

AttendanceState.cs

```
namespace TesinaMobileCloud.Shared
{
  public enum AttendanceState
  {
    Danger = 1,
    Precaution = 2,
    Good = 3
  }
}
```

ServiceModule.cs

```
using Microsoft.WindowsAzure;
using Ninject.Modules;
using TesinaMobileCloud.Data;
```

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Shared
{
   public class ServiceModule : NinjectModule
   {
      public override void Load()
      {
         Bind<CloudStorageAccount>().ToMethod(context =>
CloudStorageConfiguration.GetCloudAccount("DataConnectionString"));

         Bind<ITable<Course>>().To<AzureTable<Course>>();
         Bind<ITable<Career>>().To<AzureTable<Career>>();

Bind<ITable<CareerCourseRelationship>>().To<AzureTable<CareerCourseRelationship>>(
);
         Bind<ITable<Student>>().To<AzureTable<Student>>();

Bind<ITable<StudentCourseRelationship>>().To<AzureTable<StudentCourseRelationship>
>();


Bind<IQueue<QueueNotificationMessage>>().To<NotificationQueue<QueueNotificationMe
ssage>>();

         Bind<ICareerRepository>().To<CareerRepository>();
         Bind<ICourseRepository>().To<CourseRepository>();

Bind<ICourseCareerRelationshipRepository>().To<CareerCourseRelationshipRepository>()
;
         Bind<IStudentRepository>().To<StudentRepository>();
         Bind<IStudentCourseRepository>().To<StudentCourseRepository>();
      }
   }
}
```

4. Anexo 4 – Iteración 4

ServiceRoleTest

IServiceRoleFixture.cs

```
using System;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using Course = TesinaMobileCloud.Data.DTO.Course;
using CourseAzure = TesinaMobileCloud.Data.Model.Course;
using Student = TesinaMobileCloud.Data.Model.Student;
using TesinaMobileCloud.Data.DTO;

namespace ServiceRoleTest
{
   [TestClass]
   public class ServiceRoleFixture
   {
      const string CareerCode = "502";
      const string StudentCode = "10004";
      const int CourseCode = 120;

      private Mock<ICourseRepository> _repositoryCourse;
      private Mock<ICareerRepository> _repositoryCareer;
      private Mock<ICourseCareerRelationshipRepository> _repositoryRelationship;
      private Mock<IStudentRepository> _student;
      private Mock<IStudentCourseRepository> _studentCourse;

      [TestInitialize]
      public void Setup()
      {
         _repositoryCourse = new Mock<ICourseRepository>();
         _repositoryCareer = new Mock<ICareerRepository>();
         _repositoryRelationship = new Mock<ICourseCareerRelationshipRepository>();
         _repositoryCourse.Setup(m => m.GetCoursesByCareer(CareerCode)).Returns(new
List<TesinaMobileCloud.Data.Model.Course>
                                                {
                                                   new
TesinaMobileCloud.Data.Model.Course(CourseCode,"Programación I")
                                                   {
                                                      Professor = "Mg. Angeleri, Paula",
                                                   }
                                                });
         _repositoryRelationship.Setup(m =>
m.GetCareerYearOfCourseByCareer(int.Parse(CareerCode),
                                                   CourseCode))
                                .Returns(1);
         _student = new Mock<IStudentRepository>();
         _studentCourse = new Mock<IStudentCourseRepository>();
```

```
        }


        [TestMethod]
        public void RetriveStudentByStudentCodeAndCareer()
        {
            _student.Setup(m =>
m.GetStudentByCareerCodeAndStudentCode(It.IsAny<string>(), It.IsAny<string>()))
                    .Returns(new Student
                    {
                        CareerCode = 502,
                        StudentCode = 10004,
                        FirstName = "Carlos",
                        LastName = "Rodrigo"
                    });
            _studentCourse.Setup(m => m.RetriveCoursesOfStudent(It.IsAny<string>()))
                        .Returns(new List<StudentCourseRelationship>
                        {
                            new StudentCourseRelationship
                            {
                                CourseCode = 1,
                            }
                        });
            _repositoryCourse.Setup(m => m.GetSingleCourse(It.IsAny<int>()))
                        .Returns(new CourseAzure
                        {
                            Code = 1,
                            Title = "Ing. del Software",
                            Professor = "Mg. Paula Angeleri",
                            TotalHoursOfClasses = 60,
                            Description = "Lalo",
                            Scheduled = "Miercoles 17Hs",
                            FinalExamDate = new DateTime(2013, 12, 22),
                            PartialExameDate = new DateTime(2013, 10, 10),
                            PracticWorkPresentationDate = new DateTime(2013, 12, 20),
                            RecuperationExameDate = new DateTime(2013, 11, 20)
                        });
            _repositoryCareer.Setup(m => m.GetCareer(It.IsAny<string>())).Returns(new
TesinaMobileCloud.Data.Model.Career());

            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object, _student.Object, _studentCourse.Object,
_repositoryCareer.Object);

            var result = sut.RetriveStudentByCodeAndCareer(StudentCode, CareerCode);

            Assert.IsNotNull(result);
            Assert.IsNotNull(result.Courses);
            Assert.AreEqual(1, result.Courses.Count());
            Assert.IsNotNull(result.Courses.First().PartialExamResult);
            Assert.IsNotNull(result.Courses.First().FinalExamResult);
            Assert.IsNotNull(result.Courses.First().RecuperationExamResult);
            Assert.IsNotNull(result.Courses.First().PracticWorkResult);
            Assert.IsNotNull(result.Courses.First().TotalHoursAssisted);
```

```
            Assert.IsNotNull(result.Courses.First().Course.TotalHoursOfClasses);
            Assert.IsNotNull(result.Courses.First().Course.Title);
            Assert.IsNotNull(result.Courses.First().Course.Professor);
            Assert.IsNotNull(result.Courses.First().Course.Code);
            Assert.IsNotNull(result.Courses.First().Course.Scheduled);
            Assert.IsNotNull(result.Courses.First().Course.Description);
            Assert.IsNotNull(result.Courses.First().Course.PartialExamDate);
            Assert.IsNotNull(result.Courses.First().Course.FinalExamDate);
            Assert.IsNotNull(result.Courses.First().Course.RecuperationExameDate);
            Assert.IsNotNull(result.Courses.First().Course.PracticWorkPresentationDate);
            Assert.IsNotNull(result.Career);
        }

        [TestMethod]
        public void RetriveStudentCourse()
        {
            var studentCode ="10004";
            var courseCode = "22";
            var careerCode = "502";
            _studentCourse.Setup(m => m.RetriveStudentCourseByCode(It.IsAny<string>(),
It.IsAny<string>()))
                        .Returns(
                              new StudentCourseRelationship
                                 {
                                    CourseCode = 1,
                                 }
                          );
            _repositoryCourse.Setup(m => m.GetSingleCourse(It.IsAny<int>()))
                         .Returns(new CourseAzure
                         {
                            Code = 1,
                            Title = "Ing. del Software",
                            Professor = "Mg. Paula Angeleri",
                            TotalHoursOfClasses = 60,
                            Description = "Lalo",
                            Scheduled = "Miercoles 17Hs",
                            FinalExamDate = new DateTime(2013, 12, 22),
                            PartialExameDate = new DateTime(2013, 10, 10),
                            PracticWorkPresentationDate = new DateTime(2013, 12, 20),
                            RecuperationExameDate = new DateTime(2013, 11, 20)
                         });

            var sut = new SiaService.SiaService(_repositoryCourse.Object,
                                        _repositoryRelationship.Object,
                                        _student.Object, _studentCourse.Object,
                                        _repositoryCareer.Object);

            var result = sut.RetriveStudentCourse(careerCode, studentCode, courseCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(StudentCourse));
        }
    }
}
```

SiaService

AzureLocalStorageTraceListener.cs

```
using System;
using System.Diagnostics;
using System.IO;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class AzureLocalStorageTraceListener : XmlWriterTraceListener
    {
        public AzureLocalStorageTraceListener()
            : base(Path.Combine(AzureLocalStorageTraceListener.GetLogDirectory().Path,
"SiaService.svclog"))
        {
        }

        public static DirectoryConfiguration GetLogDirectory()
        {
            var directory = new DirectoryConfiguration
                                        {
                                            Container = "wad-tracefiles",
                                            DirectoryQuotaInMB = 10,
                                            Path =

RoleEnvironment.GetLocalResource("SiaService.svclog").RootPath
                                        };
            return directory;
        }
    }
}
```

Global.asax.cs

```
using System;
using Ninject;
using Ninject.Web.Common;
using TesinaMobileCloud.Shared;

namespace SiaService
{
    public class Global : NinjectHttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            Application_Start();
        }

        protected override IKernel CreateKernel()
        {
```

```
            return new StandardKernel(new ServiceModule());
        }
    }
```

ISiaService.cs

```
using System;
using System.ServiceModel;
using System.ServiceModel.Web;
using TesinaMobileCloud.Data.DTO;
using Student = TesinaMobileCloud.Data.DTO.Student;

namespace SiaService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "ISiaService" in both code and config file together.
    [ServiceContract]
    public interface ISiaService
    {

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RetriveStudentByCodeAndCareer/{studentCode}/{careerCode}")]
        Student RetriveStudentByCodeAndCareer(string studentCode, string careerCode);

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RetriveStudentCourse/{careerCode}/{studentCode}/{courseCode}")]
        StudentCourse RetriveStudentCourse(string careerCode, string studentCode, string
courseCode);

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RegisterPushNotificationChannelForStudent?studentCode={studentCode}&uri={uri}")]
        Student RegisterPushNotificationChannelForStudent(string studentCode, string uri);
    }
}
```

Service1.svc.cs

```
using System;
using System.IO;
using System.Runtime.Serialization;
using System.Text;
using System.Web;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using CareerDto = TesinaMobileCloud.Data.DTO.Career;
using Course = TesinaMobileCloud.Data.DTO.Course;
using Student = TesinaMobileCloud.Data.DTO.Student;
```

```
namespace SiaService
{
    public class SiaService : ISiaService
    {
        private readonly ICareerRepository _careerRepository;
        private readonly ICourseRepository _courseRepository;
        private readonly ICourseCareerRelationshipRepository
_courseCareerRelationshipRepository;
        private readonly IStudentRepository _studentRepository;
        private readonly IStudentCourseRepository _studentCourseRelationshipRepository;

        public SiaService(ICourseRepository courseRepository,
ICourseCareerRelationshipRepository courseCareerRelationshipRepository,
IStudentRepository studentRepository, IStudentCourseRepository
studentCourseRelationshipRepository, ICareerRepository careerRepository)
        {
            _courseRepository = courseRepository;
            _courseCareerRelationshipRepository = courseCareerRelationshipRepository;
            _studentRepository = studentRepository;
            _studentCourseRelationshipRepository = studentCourseRelationshipRepository;
            _careerRepository = careerRepository;
        }

        public Student RetriveStudentByCodeAndCareer(string studentCode, string
careerCode)
        {
            var studentAzure =
_studentRepository.GetStudentByCareerCodeAndStudentCode(studentCode, careerCode);
            var studentToRetrive = new Student
                {
                    FirstName = studentAzure.FirstName,
                    LastName = studentAzure.LastName,
                    StudentCode = studentAzure.StudentCode,
                    CareerCode = studentAzure.CareerCode,
                    DeviceUri = studentAzure.DeviceUri
                };
            var coursesOfStudent =
_studentCourseRelationshipRepository.RetriveCoursesOfStudent(studentCode);
            foreach (var studentCourseRelationship in coursesOfStudent)
            {
                var singleCourse =
_courseRepository.GetSingleCourse(studentCourseRelationship.CourseCode);
                studentToRetrive.Courses.Add(BuildStudentCourseDto(careerCode,
studentCourseRelationship, singleCourse));
            }

            var career = _careerRepository.GetCareer(careerCode);
            studentToRetrive.Career = new CareerDto
                {
                    Code = career.CareerCode.ToString(),
                    Title = career.Title
                };

            return studentToRetrive;
```

```
        }

        private StudentCourse BuildStudentCourseDto(string careerCode,
StudentCourseRelationship studentCourseRelationship,
TesinaMobileCloud.Data.Model.Course singleCourse)
        {
            return new StudentCourse
                {
                    FinalExamResult = studentCourseRelationship.FinalExamResult,
                    PartialExamResult = studentCourseRelationship.PartialExamResult,
                    PracticWorkResult = studentCourseRelationship.PracticWorkResult,
                    RecuperationExamResult =
studentCourseRelationship.RecuperationExamResult,
                    TotalHoursAssisted = studentCourseRelationship.TotalHoursAssisted,
                    FinalExamInscripted = studentCourseRelationship.FinalExamInscripted,
                    Course = new Course
                        {
                            Code = singleCourse.Code,
                            Professor = singleCourse.Professor,
                            Title = singleCourse.Title,
                            FinalExamDate = singleCourse.FinalExamDate,
                            PartialExamDate = singleCourse.PartialExameDate,
                            PracticWorkPresentationDate =
singleCourse.PracticWorkPresentationDate,
                            RecuperationExameDate = singleCourse.RecuperationExameDate,
                            Scheduled = singleCourse.Scheduled,
                            Description = singleCourse.Description,
                            YearOfCareer =
_courseCareerRelationshipRepository.GetCareerYearOfCourseByCareer(int.Parse(career
Code), singleCourse.Code),
                            TotalHoursOfClasses = singleCourse.TotalHoursOfClasses
                        }
                };
        }

        public StudentCourse RetriveStudentCourse(string careerCode, string studentCode,
string courseCode)
        {
            var studentCourse =
_studentCourseRelationshipRepository.RetriveStudentCourseByCode(studentCode,
courseCode);
            var course = _courseRepository.GetSingleCourse(studentCourse.CourseCode);

            return BuildStudentCourseDto(careerCode, studentCourse, course);
        }

        public Student RegisterPushNotificationChannelForStudent(string studentCode, string
uri)
        {

            var student = _studentRepository.GetStudentByStudentCode(studentCode);
            student.DeviceUri = HttpUtility.UrlDecode(uri);
            _studentRepository.Add(student);
```

```
            return new Student();
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // To enable the AzureLocalStorageTraceListner, uncomment relevent section in the
web.config
            DiagnosticMonitorConfiguration diagnosticConfig =
DiagnosticMonitor.GetDefaultInitialConfiguration();
            diagnosticConfig.Directories.ScheduledTransferPeriod =
TimeSpan.FromMinutes(1);

diagnosticConfig.Directories.DataSources.Add(AzureLocalStorageTraceListener.GetLogDir
ectory());

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

TesinaMobileCloud.AdminSite

Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
```

```
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.AdminSite
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

BundleConfig.cs

```
using System.Web;
using System.Web.Optimization;

namespace TesinaMobileCloud.AdminSite
{
    public class BundleConfig
    {
        // For more information on Bundling, visit
http://go.microsoft.com/fwlink/?LinkId=254725
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                    "~/Scripts/jquery-{version}.js"));
```

```
            bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                        "~/Scripts/jquery-ui-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then,
        when you're
            // ready for production, use the build tool at http://modernizr.com to pick only the
        tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

            bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
                        "~/Content/themes/base/jquery.ui.core.css",
                        "~/Content/themes/base/jquery.ui.resizable.css",
                        "~/Content/themes/base/jquery.ui.selectable.css",
                        "~/Content/themes/base/jquery.ui.accordion.css",
                        "~/Content/themes/base/jquery.ui.autocomplete.css",
                        "~/Content/themes/base/jquery.ui.button.css",
                        "~/Content/themes/base/jquery.ui.dialog.css",
                        "~/Content/themes/base/jquery.ui.slider.css",
                        "~/Content/themes/base/jquery.ui.tabs.css",
                        "~/Content/themes/base/jquery.ui.datepicker.css",
                        "~/Content/themes/base/jquery.ui.progressbar.css",
                        "~/Content/themes/base/jquery.ui.theme.css"));
        }
    }
}
```

FilterConfig.cs

```
using System.Web;
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

NinjectWebCommon.cs

```
using Ninject.Modules;
using TesinaMobileCloud.Shared;
```

```
[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NinjectWebCommon), "Start")]
[assembly:
WebActivator.ApplicationShutdownMethodAttribute(typeof(TesinaMobileCloud.AdminSite.A
pp_Start.NinjectWebCommon), "Stop")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;

    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var modules = new INinjectModule[] { new ServiceModule() };
            var kernel = new StandardKernel(modules);
            kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new Bootstrapper().Kernel);
            kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

            RegisterServices(kernel);
            return kernel;
        }
```

```
        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
}
```

NotificationManagementUi.cs

```
[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NotificationManagementUi), "PreStart")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System.Reflection;
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using TesinaMobileCloud.AdminSite.Areas.Notifications.Helpers;

    public static class NotificationManagementUi
    {
        public const string TileImagesContainerName = "tileimagescontainer";

        public static void PreStart()
        {
            NotificationsManagementUiContext.Configure = c =>
                {
                    // TODO: Replace with your own Windows Azure Storage account name and
key, or read it from a configuration file
                    c.CloudStorageAccount = CloudStorageAccount.DevelopmentStorageAccount;

                    // TODO: Replace with your prefered container name to store tile images
                    c.TileImagesContainerName = TileImagesContainerName;

                    return c;
                };


UploadTileImage("TesinaMobileCloud.AdminSite.Resources.WindowsAzureLogo.png");

UploadTileImage("TesinaMobileCloud.AdminSite.Resources.WindowsPhoneLogo.png");

UploadTileImage("TesinaMobileCloud.AdminSite.Resources.AzureBackground.png");

UploadTileImage("TesinaMobileCloud.AdminSite.Resources.DefaultBackground.png");
        }

        private static void UploadTileImage(string imageName)
        {
```

```
            var cloudBlobClient =
NotificationsManagementUiContext.Current.CloudStorageAccount.CreateCloudBlobClient()
;
            var tileImagesContainerName =
NotificationsManagementUiContext.Current.TileImagesContainerName;

            // Create the container (and make it public)
            var container = cloudBlobClient.GetContainerReference(tileImagesContainerName);
            container.CreateIfNotExist();
            container.SetPermissions(
                new BlobContainerPermissions
                {
                    PublicAccess = BlobContainerPublicAccessType.Blob
                });

            // Upload the image from the assembly resources.
            var assembly = Assembly.GetExecutingAssembly();
            var imageStream = assembly.GetManifestResourceStream(imageName);
            var blob =
container.GetBlobReference(imageName.Replace("TesinaMobileCloud.AdminSite.Resourc
es.", string.Empty));
            blob.Properties.ContentType = "image/png";
            blob.UploadFromStream(imageStream);
        }
    }
}
```

RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

WebApiConfig.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace TesinaMobileCloud.AdminSite
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Controllers

CareerController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerController : Controller
    {
        private readonly ITable<Career> _careers;

        public CareerController(ITable<Career> careers)
        {
            _careers = careers;
        }

//
        // GET: /Career/

        public ActionResult Index()
        {
            return View(_careers.GetByCriteria(career =>
!string.IsNullOrEmpty(career.PartitionKey)));
        }

        //
        // GET: /Career/Details/5
```

```
        public ActionResult Details(int id)
        {
            return View(_careers.GetSingleByCriteria(career => career.CareerCode == id));
        }

        //
        // GET: /Career/Create

        public ActionResult Create()
        {
            return View(new Career());
        }

        //
        // POST: /Career/Create

        [HttpPost]
        public ActionResult Create(Career career)
        {
            try
            {
                _careers.Add(career);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_careers.GetSingleByCriteria(career1 => career1.CareerCode == id));
        }

        //
        // POST: /Career/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var career = _careers.GetSingleByCriteria(career1 => career1.CareerCode ==
id);
                UpdateModel(career);
                _careers.Add(career);

                return RedirectToAction("Index");
            }
            catch
```

```
            {
                return View();
            }
        }

        //
        // GET: /Career/Delete/5

        public ActionResult Delete(int id)
        {
            return View();
        }

        //
        // POST: /Career/Delete/5

        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                _careers.Delete(_careers.GetSingleByCriteria(c => c.CareerCode == id));

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CareerCourseRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerCourseRelationshipController : Controller
    {
        private readonly ITable<CareerCourseRelationship> _careerCourseRelationship;

        public CareerCourseRelationshipController(ITable<CareerCourseRelationship>
careerCourseRelationship)
        {
            _careerCourseRelationship = careerCourseRelationship;
        }
```

```
//
        // GET: /CareerCourseRelationship/

        public ActionResult Index()
        {
            return View(_careerCourseRelationship.GetByCriteria(relationship =>
relationship.CareerCode != 0 ));
        }

        //
        // GET: /CareerCourseRelationship/Details/5

        public ActionResult Details(int course, int career)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CourseCode == course && relationship.CareerCode == career));
        }

        //
        // GET: /CareerCourseRelationship/Create

        public ActionResult Create()
        {
            return View(new CareerCourseRelationship());
        }

        //
        // POST: /CareerCourseRelationship/Create

        [HttpPost]
        public ActionResult Create(CareerCourseRelationship careerCourseRelationship)
        {
            try
            {
                _careerCourseRelationship.Add(careerCourseRelationship);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /CareerCourseRelationship/Edit/5

        public ActionResult Edit(int course, int career)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CourseCode == course && relationship.CareerCode == career));
        }

        //
```

```
        // POST: /CareerCourseRelationship/Edit/5

        [HttpPost]
        public ActionResult Edit(int careerCode, int courseCode, FormCollection collection)
        {
            try
            {
                var courseInCloud = _careerCourseRelationship.GetSingleByCriteria(s =>
s.CareerCode == careerCode && s.CourseCode == courseCode);
                UpdateModel(courseInCloud);
                _careerCourseRelationship.Add(courseInCloud);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /CareerCourseRelationship/Delete/5

        public ActionResult Delete(int careerCode, int courseCode)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(s => s.CareerCode ==
careerCode && s.CourseCode == courseCode));
        }

        //
        // POST: /CareerCourseRelationship/Delete/5

        [HttpPost]
        public ActionResult Delete(int careerCode, int courseCode, FormCollection collection)
        {
            try
            {

_careerCourseRelationship.Delete(_careerCourseRelationship.GetSingleByCriteria(c =>
c.CareerCode == careerCode && c.CourseCode == courseCode));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CourseController.cs

```
using System;
using System.Web.Mvc;
```

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CourseController : Controller
    {
        private readonly ITable<Course> _courses;

        public CourseController(ITable<Course> courses)
        {
            _courses = courses;
        }

        public ActionResult Index()
        {
            return View(_courses.GetByCriteria(course =>
!string.IsNullOrEmpty(course.PartitionKey)));
        }

        //
        // GET: /Course/Details/5

        public ActionResult Details(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.Code == id));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new Course(0, string.Empty));
        }

        //
        // POST: /Course/Create

        [HttpPost]
        public ActionResult Create(Course course)
        {
            try
            {
                _courses.Add(course);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
```

```
        // GET: /Course/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.Code == id));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _courses.GetSingleByCriteria(course => course.Code ==
id);
                UpdateModel(courseInCloud);
                _courses.Add(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id)
        {
            try
            {
                _courses.Delete( _courses.GetSingleByCriteria(c => c.PartitionKey == id));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // POST: /Course/Delete/5


    }
}
```

HomeController.cs

```
using System.Web.Mvc;
```

```
namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Sitio de administración y soporte, Tesina Carlos Rodrigo";

            return View();
        }
    }
}
```

StudentCodeRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Web;
using System.Web.Helpers;
using System.Web.Mvc;
using Newtonsoft.Json;
using TesinaMobileCloud.Data.Helpers;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Data.Table;
using TesinaMobileCloud.Notifications;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class StudentCodeRelationshipController : Controller
    {
        private readonly ITable<StudentCourseRelationship> _table;
        private readonly ITable<Course> _courses;
        private readonly ITable<Student> _students;
        private readonly IQueue<QueueNotificationMessage> _queueNotifications;

        public StudentCodeRelationshipController(ITable<StudentCourseRelationship> table,
ITable<Course> courses, IQueue<QueueNotificationMessage> queueNotifications,
ITable<Student> students)
        {
            _table = table;
            _courses = courses;
            _queueNotifications = queueNotifications;
            _students = students;
        }

        public ActionResult Index()
        {
            return View(_table.GetByCriteria(relationship => relationship.StudentCode != 0));
        }

        //
        // GET: /StudentCodeRelationship/Details/5
```

```
        public ActionResult Details(int id, int courseCode)
        {
            return View(_table.GetSingleByCriteria(s => s.StudentCode == id && s.CourseCode
== courseCode));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new StudentCourseRelationship());
        }

        //
        // POST: /Course/Create

        [HttpPost]
        public ActionResult Create(StudentCourseRelationship student)
        {
            try
            {
                _table.Add(student);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id, int courseCode)
        {
            return View(_table.GetSingleByCriteria(s => s.StudentCode == id && s.CourseCode
== courseCode));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, int courseCode, FormCollection collection)
        {
            try
            {
                var courseInCloud = _table.GetSingleByCriteria(s => s.StudentCode == id &&
s.CourseCode == courseCode);
                UpdateModel(courseInCloud);
                _table.Add(courseInCloud);
                return RedirectToAction("Index");
```

```
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
                _table.Delete(_table.GetSingleByCriteria(c => c.PartitionKey == id && c.RowKey
== row));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        public ActionResult ExamResultAsign(int id, int course)
        {
            try
            {
                ViewBag.ExamTypes =
                    new List<SelectListItem>
                    {
                        new SelectListItem {Text = "Parcial", Value = "PartialExam"},
                        new SelectListItem {Text = "Recuperatorio", Value =
"RecuperationExam"},
                        new SelectListItem {Text = "Final", Value = "FinalExam"},
                        new SelectListItem
                        {
                            Text = "Presentación de Trabajos Practicos",
                            Value = "PracticalWorkExam"
                        },
                    };

                return View(_table.GetSingleByCriteria(s => s.StudentCode == id &&
s.CourseCode == course));
            }
            catch
            {
                return View();
            }
        }

        [HttpPost]
        public ActionResult ExamResultAsign(int studentCode, int courseCode,
FormCollection form)
```

```
      {
         var selectListItems = new List<SelectListItem>
            {
               new SelectListItem {Text = "Parcial", Value = "PartialExam"},
               new SelectListItem {Text = "Recuperatorio", Value = "RecuperationExam"},
               new SelectListItem {Text = "Final", Value = "FinalExam"},
               new SelectListItem {Text = "Presentación de Trabajos Practicos", Value =
"PracticalWorkExam"},
            };
         try
         {
            var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode ==
studentCode && sc.CourseCode == courseCode);
            var course = _courses.GetSingleByCriteria(c => c.Code == courseCode);
            var student = _students.GetSingleByCriteria(s => s.StudentCode ==
studentCode);
            var examType = form["SelectedExamType"];
            var examResult = int.Parse(form["ExamResultText"]);

            if (examResult > 10 || examResult == 0)
            {
               ModelState.AddModelError(string.Empty, "Calificación no valida");
               ViewBag.ExamTypes =
                selectListItems;
               return View(_table.GetSingleByCriteria(s => s.StudentCode == studentCode
&& s.CourseCode == courseCode));
            }



            var tileNotification = new TileNotification
            {
               BackBackgroundImage = "Void"
            };

            var toastNotification = new ToastNotification
               {
                  Title = course.Title,
                  Page =
String.Format("/View/CourseView.xaml?CareerCode={0}&amp;StudentCode={1}&amp;Cou
rseCode={2}",
                                 student.CareerCode,
                                 student.StudentCode,
                                 courseCode)
               };

            switch (examType)
            {
               case "PartialExam":
                  studentCourse.PartialExamResult = examResult;
                  var partialExamResult = String.Format("{1}. Examen Parcial: {0}",
examResult, course.Title);
                  tileNotification.BackContent = partialExamResult;
                  toastNotification.Content = partialExamResult;
```

```
                    break;
                case "RecuperationExam":
                    studentCourse.RecuperationExamResult = examResult;
                    var recuperationExamResult = String.Format("{1}. Examen Recuperatorio:
{0}", examResult, course.Title);
                    tileNotification.BackContent = recuperationExamResult;
                    toastNotification.Content = recuperationExamResult;
                    break;
                case "FinalExam":
                    studentCourse.FinalExamResult = examResult;
                    var finalExamResult = String.Format("{1}. Examen Final: {0}", examResult,
course.Title);
                    tileNotification.BackContent = finalExamResult;
                    toastNotification.Content = finalExamResult;
                    break;
                default:
                    studentCourse.PracticWorkResult = examResult;
                    var tpResult = String.Format("{1}. Presentación TPs: {0}", examResult,
course.Title);
                    tileNotification.BackContent = tpResult;
                    toastNotification.Content = tpResult;
                    break;
            }

            _table.Add(studentCourse);

            _queueNotifications.AddMessage(new QueueNotificationMessage
                {
                    Destinatary = student.DeviceUri,
                    Type = tileNotification.Type,
                    Payload = tileNotification.Payload
                });

            _queueNotifications.AddMessage(new QueueNotificationMessage
            {
                Destinatary = student.DeviceUri,
                Type = toastNotification.Type,
                Payload = toastNotification.Payload
            });

            return RedirectToAction("Index");
        }
        catch (Exception)
        {
            ViewBag.ExamTypes =
                selectListItems;
            return View(_table.GetSingleByCriteria(s => s.StudentCode == studentCode &&
s.CourseCode == courseCode));
        }
    }

    public ActionResult AsignAssistedHours(int id, int course)
    {
```

```
            var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode == id &&
sc.CourseCode == course);
            return View(studentCourse);
        }

        [HttpPost]
        public ActionResult AsignAssistedHours(int studentCode, int courseCode,
FormCollection collection)
        {
            var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode ==
studentCode && sc.CourseCode == courseCode);
            var course = _courses.GetSingleByCriteria(c => c.Code == courseCode);
            var studentDeviceUri = _students.GetSingleByCriteria(s => s.StudentCode ==
studentCode).DeviceUri;

            var previousAssistanceState =
studentCourse.GetAttendanceState(course.TotalHoursOfClasses, course.IsCritical);
            try
            {
                var hours = int.Parse(collection["AddHours"]);

                studentCourse.TotalHoursAssisted += hours;
                _table.Add(studentCourse);

                AddAttedanceChangeNotification(studentCourse, course, studentDeviceUri,
previousAssistanceState);

                return RedirectToAction("Index");
            }
            catch (Exception)
            {
                return View(studentCourse);
            }
        }

        private void AddAttedanceChangeNotification(StudentCourseRelationship
studentCourse, Course course, string studentDeviceUri, AttendanceState
previousAssistanceState)
        {
            if (!string.IsNullOrEmpty(studentDeviceUri))
            {
                var actualAssistanceState =
studentCourse.GetAttendanceState(course.TotalHoursOfClasses, course.IsCritical);
                if (previousAssistanceState != actualAssistanceState)
                {
                    var tileNotification = new TileNotification
                    {
                        BackTitle = string.Format("Asistencia: {0}",
GetActualAssistanceStateText(actualAssistanceState)),
                        BackContent = course.Title,
                        BackBackgroundImage = actualAssistanceState.ToString()
                    };

                    _queueNotifications.AddMessage(new QueueNotificationMessage
```

```
                {
                    Destinatary = studentDeviceUri,
                    Type = tileNotification.Type,
                    Payload = tileNotification.Payload
                });
            }
        }
    }

    private static string GetActualAssistanceStateText(AttendanceState
actualAssistanceState)
    {
        switch (actualAssistanceState)
        {
            case AttendanceState.Danger:
                return "Baja";
            case AttendanceState.Precaution:
                return "Cuidado";
            default:
                return "Buena";
        }
    }
  }
}
```

StudentController.cs

```
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class StudentController : Controller
    {
        private readonly ITable<Student> _students;

        public StudentController(ITable<Student> students)
        {
            _students = students;
        }

        public ActionResult Index()
        {
            return View(_students.GetByCriteria(s => !string.IsNullOrEmpty(s.PartitionKey)));
        }

        public ActionResult Details(int id)
        {
            return View(_students.GetSingleByCriteria(s => s.StudentCode == id));
        }

        //
        // GET: /Course/Create
```

```
public ActionResult Create()
{
    return View(new Student());
}

//
// POST: /Course/Create

[HttpPost]
public ActionResult Create(Student student)
{
    try
    {
        _students.Add(student);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /Course/Edit/5

public ActionResult Edit(int id)
{
    return View(_students.GetSingleByCriteria(s => s.StudentCode == id));
}

//
// POST: /Course/Edit/5

[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
    try
    {
        var courseInCloud = _students.GetSingleByCriteria(s => s.StudentCode == id);
        UpdateModel(courseInCloud);
        _students.Add(courseInCloud);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /Course/Delete/5

public ActionResult Delete(string id, string row)
```

```
        {
            try
            {
                _students.Delete(_students.GetSingleByCriteria(c => c.PartitionKey == id &&
c.RowKey == row));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

Views

_ViewStart.cshtml

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

C:\Users\Carlos\SkyDrive\Tesina\Tesina Source Code\TesinaMobileCloud\TesinaMobileCloud.AdminSite\Views\Career Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Nueva";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Dar de Alta Carrera</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
```

```
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Career

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Eliminar</h2>

<h3>Está seguro de eliminar esta Carrera?</h3>
<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
@using (Html.BeginForm())
{
    <p>
        <input type="submit" value="Eliminar" />
        |
        @Html.ActionLink("Volver", "Index")
    </p>
```

```
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { id=Model.CareerCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Editar";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código")
```

```
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Career>

@{
    ViewBag.Title = "Carreras";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Carreras</h2>

<p>
    @Html.ActionLink("Nueva", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("CareerCode", "Código")
        </th>
        <th>
            @Html.Label("Title", "Título")
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
```

```
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { id = item.CareerCode }) |
            @Html.ActionLink("Detalles", "Details", new { id = item.CareerCode }) |
            @Html.ActionLink("Eliminar", "Delete", new { id = item.CareerCode })
        </td>
    </tr>
}

</table>
```

CareerCourseRelationship

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Asignar Cátedra a Carrera</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de la Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de la Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("YearInTheCareer", "Año de Cursada")
```

```
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>


        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar Relación</h2>

<h3>Está seguro que desea eliminar esta relación?</h3>
<fieldset>
    <legend>Relación Cátedra-Carrera</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("YearInTheCareer", "Año en que se dicta")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.YearInTheCareer)
```

```
    </div>

</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Relación Cátedra-Carrera</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("YearInTheCareer", "Año en que se dicta")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.YearInTheCareer)
    </div>

</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { course = Model.CourseCode, career =
Model.CareerCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship
```

```
@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Relación Cátedra-Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("YearInTheCareer", "Año en que se dicta")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.CareerCourseRelationship>
```

```
@{
    ViewBag.Title = "Asignar Cátedra a Carrera";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Asignación de Cátedras a Carreras</h2>

<p>
    @Html.ActionLink("Nueva asignación", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("CareerCode", "Código de Carrera")
        </th>
        <th>
            @Html.Label("CourseCode", "Código de Cátedra")
        </th>
        <th>
            @Html.Label("YearInTheCareer", "Año en que se dicta")
        </th>

        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CourseCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.YearInTheCareer)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new     { course = item.CourseCode, career =
item.CareerCode }) |
            @Html.ActionLink("Detalles", "Details",new { course = item.CourseCode, career =
item.CareerCode }) |
            @Html.ActionLink("Eliminar", "Delete", new { courseCode = item.CourseCode,
careerCode = item.CareerCode } )
        </td>
    </tr>
}

</table>
```

Course

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Nuevo";
}

<h2>Dar de Alta Cátedra</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Cátedra</legend>

        <div class="editor-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Code)
            @Html.ValidationMessageFor(model => model.Code)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.Label("Professor", "Docente")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>
        <div class="editor-label">
            @Html.Label("FinalExamDate", "Fecha de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamDate)
            @Html.ValidationMessageFor(model => model.FinalExamDate)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExameDate)
```

```
            @Html.ValidationMessageFor(model => model.PartialExameDate)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkPresentationDate)
            @Html.ValidationMessageFor(model => model.PracticWorkPresentationDate)
        </div>
        <div class="editor-label">
            @Html.Label("RecuperationExameDate", "Fecha de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExameDate)
            @Html.ValidationMessageFor(model => model.RecuperationExameDate)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>

        <div class="editor-label">
            @Html.Label("Scheduled", "Horario")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Scheduled)
            @Html.ValidationMessageFor(model => model.Scheduled)
        </div>

        <div class="editor-label">
            @Html.Label("Description", "Descripción de la Cátedra")
        </div>
        <div class="editor-field">
            @Html.TextAreaFor(model => model.Description)
            @Html.ValidationMessageFor(model => model.Description)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar</h2>

<h3>Está seguro que desea eliminar la Cátedra?</h3>
<fieldset>
    <legend>Course</legend>

    <div class="display-label">
        @Html.Label("Code", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Code)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>

    <div class="display-label">
        @Html.Label("Professor", "Docente")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Detalles";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Cátedra</legend>
```

```
    <div class="display-label">
        @Html.Label("Code", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Code)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>

    <div class="display-label">
        @Html.Label("Professor", "Docente")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
    <div class="editor-label">
        @Html.Label("FinalExamDate", "Fecha de Examen Final")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.FinalExamDate)
    </div>

    <div class="editor-label">
        @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.PartialExameDate)
    </div>

    <div class="editor-label">
        @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.PracticWorkPresentationDate)
    </div>
    <div class="editor-label">
        @Html.Label("RecuperationExame", "Fecha de Examen Recuperatorio")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.RecuperationExameDate)
    </div>
    <div class="editor-label">
        @Html.Label("TotalHoursOfClasses", "Total de Horas de Cursada")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.TotalHoursOfClasses)
    </div>
```

```
    <div class="editor-label">
        @Html.Label("Scheduled", "Horario")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.Scheduled)
    </div>

    <div class="editor-label">
        @Html.Label("Description", "Descripción de la Cátedra")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.Description)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { id = Model.Code }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Cátedra</legend>

        <div class="editor-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Code)
            @Html.ValidationMessageFor(model => model.Code)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
```

```
            @Html.Label("Professor", "Docente")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>

       <div class="editor-label">
            @Html.Label("FinalExamDate", "Fecha de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamDate)
            @Html.ValidationMessageFor(model => model.FinalExamDate)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExameDate)
            @Html.ValidationMessageFor(model => model.PartialExameDate)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkPresentationDate)
            @Html.ValidationMessageFor(model => model.PracticWorkPresentationDate)
        </div>
        <div class="editor-label">
            @Html.Label("RecuperationExameDate", "Fecha de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExameDate)
            @Html.ValidationMessageFor(model => model.RecuperationExameDate)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>

        <div class="editor-label">
            @Html.Label("Scheduled", "Horario")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Scheduled)
            @Html.ValidationMessageFor(model => model.Scheduled)
        </div>
```

```
        <div class="editor-label">
            @Html.Label("Description", "Descripción de la Cátedra")
        </div>
        <div class="editor-field">
            @Html.TextAreaFor(model => model.Description)
            @Html.ValidationMessageFor(model => model.Description)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Course>

@{
    ViewBag.Title = "Cátedras";
}
<style type="text/css">
th {
    padding-right: 10px;
}
</style>
<h2>Cátedras</h2>

<p>
    @Html.ActionLink("Nueva", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("Code", "Código")
        </th>
        <th>
            @Html.Label("Title", "Título")
        </th>
        <th>
            @Html.Label("Professor", "Docente")
        </th>
        <th>
            @Html.Label("Scheduled", "Horario")
        </th>
        <th>
```

```
            @Html.Label("TotalHoursOfClasses", "Horas de Cursada")
        </th>
        <th></th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Code)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Professor)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Scheduled)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursOfClasses)
            </td>
            <td>
                @Html.ActionLink("Editar", "Edit", new { id = item.Code  }) |
            @Html.ActionLink("Detalles", "Details", new { id = item.Code }) |
            @Html.ActionLink("Eliminar", "Delete", new { id = item.Code })
            </td>
        </tr>
    }

</table>
```

C:\Users\Carlos\SkyDrive\Tesina\Tesina Source Code\TesinaMobileCloud\TesinaMobileCloud.AdminSite\Views\Home Index.cshtml

```
@{
    ViewBag.Title = "Sitio de Administración - Tesina Carlos Rodrigo";
}
@section featured {
    <section class="featured">
        <div class="content-wrapper">
            <hgroup class="title">
                <h1>@ViewBag.Title.</h1>
            </hgroup>
            <p>
                Este sitio provee las funciones de administración y carga de datos necesarios para el desarrollo de la tesina.
                La necesidad surge de simular los sistemas de asistencias y administración de los alumnos, para solo dedicarse a
                la implementación del cliente Mobile.
            </p>
        </div>
    </section>
}
```

Shared

Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<hgroup class="title">
    <h1 class="error">Error.</h1>
    <h2 class="error">Se ha producido un error inesperado.</h2>
</hgroup>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width" />
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
                </div>
                <div class="float-right">
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("Cátedras", "Index", "Course")</li>
                            <li>@Html.ActionLink("Carreras", "Index", "Career")</li>
                            <li>@Html.ActionLink("Cátedras-Carreras", "Index",
"CareerCourseRelationship")</li>
                            <li>@Html.ActionLink("Estudiantes", "Index", "Student")</li>
                            <li>@Html.ActionLink("Estudiantes-Cátedras", "Index",
"StudentCodeRelationship")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
```

```
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
            </div>
        </footer>

        @Scripts.Render("~/bundles/jquery")
        @RenderSection("scripts", required: false)
    </body>
</html>
```

_LoginPartial.cshtml

```
@if (Request.IsAuthenticated) {
    <text>
        Hello, @Html.ActionLink(User.Identity.Name, "Manage", "Account", routeValues: null,
htmlAttributes: new { @class = "username", title = "Manage" })!
        @using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id =
"logoutForm" })) {
            @Html.AntiForgeryToken()
            <a href="javascript:document.getElementById('logoutForm').submit()">Log off</a>
        }
    </text>
} else {
    <ul>
        <li>@Html.ActionLink("Register", "Register", "Account", routeValues: null,
htmlAttributes: new { id = "registerLink" })</li>
        <li>@Html.ActionLink("Log in", "Login", "Account", routeValues: null, htmlAttributes:
new { id = "loginLink" })</li>
    </ul>
}
```

Student

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Create";
}

<h2>Nuevo Estudiante</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Estudiante</legend>
```

```
            <div class="editor-label">
                @Html.Label("StudentCode", "Código")
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.StudentCode)
                @Html.ValidationMessageFor(model => model.StudentCode)
            </div>

            <div class="editor-label">
                @Html.Label("CareerCode", "Código de Carrera")
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.CareerCode)
                @Html.ValidationMessageFor(model => model.CareerCode)
            </div>

            <div class="editor-label">
                @Html.Label("FirstName", "Nombres")
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.FirstName)
                @Html.ValidationMessageFor(model => model.FirstName)
            </div>

            <div class="editor-label">
                @Html.Label("LastName", "Apellido")
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.LastName)
                @Html.ValidationMessageFor(model => model.LastName)
            </div>
            <p>
                <input type="submit" value="Guardar" />
            </p>
        </fieldset>
    }

    <div>
        @Html.ActionLink("Volver", "Index")
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }
```

Delete.cshtml

```
    @model TesinaMobileCloud.Data.Model.Student

    @{
        ViewBag.Title = "Delete";
    }
```

```
<h2>Eliminar Estudiante</h2>

<h3>Está seguro que desea eliminar este Estudiante?</h3>
<fieldset>
    <legend>Estudiante</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("FirstName", "Nombres")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FirstName)
    </div>

    <div class="display-label">
        @Html.Label("LastName", "Apellido")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.LastName)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Estudiante</legend>
```

```
    <div class="display-label">
        @Html.Label("StudentCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("FirstName", "Nombres")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FirstName)
    </div>

    <div class="display-label">
        @Html.Label("LastName", "Apellido")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.LastName)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new {  id=Model.StudentCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Estudiante</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
```

```
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("FirstName", "Nombres")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FirstName)
            @Html.ValidationMessageFor(model => model.FirstName)
        </div>

        <div class="editor-label">
            @Html.Label("LastName", "Apellido")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.LastName)
            @Html.ValidationMessageFor(model => model.LastName)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Student>

@{
    ViewBag.Title = "Index";
}

<h2>Estudiantes</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table>
```

```
    <tr>
        <th>
            @Html.Label("CareerCode", "Carrera")
        </th>
        <th>
            @Html.Label("StudentCode", "Matricula")
        </th>
        <th>
            @Html.Label("FirstName", "Nombre")
        </th>
        <th>
            @Html.Label("LastName", "Apellido")
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.StudentCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.FirstName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.LastName)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new {  id=item.StudentCode}) |
            @Html.ActionLink("Detalles", "Details", new { id=item.StudentCode }) |
            @Html.ActionLink("Eliminar", "Delete", new { id=item.StudentCode})
        </td>
    </tr>
}

</table>
```

StudentCodeRelationship

AsignAssistedHours.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Asignar Horas Cursadas";
}

<h2>Asignar Horas Cursadas</h2>
<div>
    @using (Html.BeginForm())
    {
```

```
            <fieldset>
                <legend>Asignación</legend>
                @Html.HiddenFor(model => model.StudentCode)
                @Html.HiddenFor(model => model.CourseCode)
                <div id="ActualAssitedHours">
                    <label>Horas Asistidas a la fecha</label>
                    @Html.DisplayFor(model => model.TotalHoursAssisted)
                </div>
                <div id="AddAssitedHours">
                    <label>Agregar Horas</label>
                    @Html.TextBox("AddHours")
                </div>
                <p>
                    <input type="submit" value="Guardar" />
                </p>
            </fieldset>
        }

        <div>
            @Html.ActionLink("Volver", "Index")
        </div>
    </div>
```

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Create";
}

<h2>Asignación de Cátedras a Estudiantes</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código de Estudiante")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>
```

```
        @*<div class="editor-label">
            @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExamResult)
            @Html.ValidationMessageFor(model => model.PartialExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("FinalExamResult", "Calificación de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamResult)
            @Html.ValidationMessageFor(model => model.FinalExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExamResult)
            @Html.ValidationMessageFor(model => model.RecuperationExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkResult)
            @Html.ValidationMessageFor(model => model.PracticWorkResult)
        </div>*@

        <div class="editor-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursAssisted)
            @Html.ValidationMessageFor(model => model.TotalHoursAssisted)
        </div>
        <div class="editor-label">
            @Html.Label("FinalExamInscripted", "Inscripto en Examen Final")
        </div>
        <div class="editor-field">
            @Html.CheckBoxFor(model => model.FinalExamInscripted)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
```

```
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar Asignación</h2>

<h3>Está seguro que desea eliminar esta asignación?</h3>
<fieldset>
    <legend>Asignación de Estudiante a Cátedra</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código de Estudiante")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
    </div>
    <div class="display-field">
        @if (Model.PartialExamResult != 0)
        {
            @Html.DisplayFor(model => model.PartialExamResult)
        }
        else
        {
            <label>-</label>
        }
    </div>

    <div class="display-label">
        @Html.Label("FinalExamResult", "Calificación de Examen Final")
    </div>
    <div class="display-field">
        @if (Model.FinalExamResult != 0)
```

```
            {
                @Html.DisplayFor(model => model.FinalExamResult)
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="display-label">
            @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
        </div>
        <div class="display-field">
            @if (Model.RecuperationExamResult != 0)
            {
                @Html.DisplayFor(model => model.RecuperationExamResult)
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="display-label">
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </div>
        <div class="display-field">
            @if (Model.PracticWorkResult != 0)
            {
                @Html.DisplayFor(model => model.PracticWorkResult)
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="display-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="display-field">
            @Html.DisplayFor(model => model.TotalHoursAssisted)
        </div>

    </fieldset>
    @using (Html.BeginForm())
    {
        <p>
            <input type="submit" value="Eliminar" />
            |
            @Html.ActionLink("Volver", "Index")
        </p>
    }
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Asignación Estudiante a Cátedra</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código de Estudiante")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Cátedra")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.PartialExamResult)
    </div>

    <div class="display-label">
        @Html.Label("FinalExamResult", "Calificación de Examen Final")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FinalExamResult)
    </div>

    <div class="display-label">
        @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.RecuperationExamResult)
    </div>

    <div class="display-label">
        @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.PracticWorkResult)
    </div>
```

```
    <div class="display-label">
        @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.TotalHoursAssisted)
    </div>

</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new {  id=Model.StudentCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código de Estudiante")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Cátedra")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.PartialExamResult)
            @if (Model.PartialExamResult != 0)
            {
```

```
            @Html.DisplayFor(model => model.PracticWorkResult)
            @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "PartialExam" })
        }
        else
        {
            <label>-</label>
        }
    </div>

    <div class="editor-label">
        @Html.Label("FinalExamResult", "Calificación de Examen Final")
    </div>
    <div class="editor-field">
        @Html.HiddenFor(model => model.FinalExamResult)
        @if (Model.FinalExamResult != 0)
        {
            @Html.DisplayFor(model => model.FinalExamResult)
            @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "FinalExam" })
        }
        else
        {
            <label>-</label>
        }
    </div>

    <div class="editor-label">
        @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
    </div>
    <div class="editor-field">
        @Html.HiddenFor(model => model.RecuperationExamResult)
        @if (Model.RecuperationExamResult != 0)
        {
            @Html.DisplayFor(model => model.RecuperationExamResult)
            @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "RecuperationExam" })
        }
        else
        {
            <label>-</label>
        }
    </div>

    <div class="editor-label">
        @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
    </div>
    <div class="editor-field">
        @Html.HiddenFor(model => model.PracticWorkResult)
        @if (Model.PracticWorkResult != 0)
        {
            @Html.DisplayFor(model => model.PracticWorkResult)
            @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "PracticWorkResult" })
```

```
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="editor-field">
            @Html.DisplayFor(model => model.TotalHoursAssisted)
            @Html.ActionLink("Asignar Horas Cursadas", "AsignAssistedHours", new { id =
Model.StudentCode, course = Model.CourseCode })
        </div>
         <div class="editor-label">
            @Html.Label("FinalExamInscripted", "Inscripto en Examen Final")
        </div>
        <div class="editor-field">
            @Html.CheckBoxFor(model => model.FinalExamInscripted)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

ExamResultAsign.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Asignar Calificación de Examen";
}

<h2>Asignar Calificación de Examen</h2>
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>
        @Html.HiddenFor(model => model.StudentCode)
        @Html.HiddenFor(model => model.CourseCode)
        <div class="editor-label">
```

```
            @Html.Label("ExamType", "Examen")
        </div>
        <div class="editor-field">
            @Html.DropDownList("SelectedExamType",
(List<SelectListItem>)ViewBag.ExamTypes)
        </div>

        <div class="editor-label">
            @Html.Label("ExamResult", "Calificación")
        </div>
        <div class="editor-field">
            @Html.TextBox("ExamResultText")
        </div>
    <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.StudentCourseRelationship>

@{
    ViewBag.Title = "Index";
}

<h2>Estado del Estudiante en una Cátedra</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("StudentCode", "Matricula del Estudiante")
        </th>
        <th>
            @Html.Label("CourseCode", "Código de la Cátedra")
        </th>
        <th>
            @Html.Label("PartialExamResult", "Resultado de Examen Parcial")
        </th>
        <th>
            @Html.Label("FinalExamResult", "Resultado de Examen Final")
        </th>
        <th>
            @Html.Label("RecuperationExamResult", "Resultado de Examen Recuperatorio")
        </th>
        <th>
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
```

```
    </th>
    <th>
        @Html.Label("TotalHoursAssisted", "Total de Horas Asistidas")
    </th>
    <th></th>
</tr>

@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.StudentCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CourseCode)
        </td>
        <td>
            @if (item.PartialExamResult != 0)
            {
                @Html.DisplayFor(modelItem => item.PartialExamResult)
            }
            else
            {
                <label>-</label>
            }
        </td>
        <td>
            @if (item.FinalExamResult != 0)
            {
                @Html.DisplayFor(modelItem => item.FinalExamResult)
            }
            else
            {
                <label>-</label>
            }
        </td>
        <td>
            @if (item.RecuperationExamResult != 0)
            {
                @Html.DisplayFor(modelItem => item.RecuperationExamResult)
            }
            else
            {
                <label>-</label>
            }
        </td>
        <td>
            @if (item.PracticWorkResult != 0)
            {
                @Html.DisplayFor(modelItem => item.PracticWorkResult)
            }
            else
            {
                <label>-</label>
```

```
                }
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursAssisted)
            </td>

            <td>
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
item.StudentCode, course = item.CourseCode })
                @Html.ActionLink("Asignar Horas Cursadas", "AsignAssistedHours", new { id =
item.StudentCode, course = item.CourseCode })
                @Html.ActionLink("Editar", "Edit", new { id = item.StudentCode, courseCode =
item.CourseCode }) |
                @Html.ActionLink("Detalles", "Details", new { id = item.StudentCode, courseCode
= item.CourseCode }) |
                @Html.ActionLink("Eliminar", "Delete", new { id = item.StudentCode, row =
item.CourseCode })
            </td>
        </tr>
    }

</table>
```

TesinaMobileCloud.Data

CloudStorageConfiguration.cs

```
using System;
using System.Configuration;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.Data
{
    public class CloudStorageConfiguration
    {
        public static CloudStorageAccount GetCloudAccount(string
cloudStorageConnectionStringName)
        {
            try
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
            catch (InvalidOperationException)
            {
                return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
            }
        }

        private static CloudStorageAccount SetConfigurationAndGetAccount(string
cloudStorageConnectionStringName)
        {
            SetConfigurationSettingsPublisher();
```

```
            return
CloudStorageAccount.FromConfigurationSetting(cloudStorageConnectionStringName);
        }

        private static void SetConfigurationSettingsPublisher()
        {
            CloudStorageAccount.SetConfigurationSettingPublisher((configName,
configSettingPublisher) =>
            {
                var configValue = ConfigurationManager.AppSettings[configName];
                if (RoleEnvironment.IsAvailable)
                    configValue = RoleEnvironment.GetConfigurationSettingValue(configName);
                configSettingPublisher(configValue);
            });
        }

    }
}
```

DTO

Attendance.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Attendance
    {
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public DateTime TotalHoursOfClasses { get; set; }
        [DataMember]
        public int ActualPercentage { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Career
    {
        [DataMember]
        public string Code { get; set; }
        [DataMember]
        public string Title { get; set; }
    }
}
```

Course.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Course
    {
        [DataMember]
        public int Code { get; set; }
        [DataMember]
        public string Title { get; set; }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public int YearOfCareer { get; set; }
        [DataMember]
        public DateTime? PartialExamDate { get; set; }
        [DataMember]
        public DateTime? FinalExamDate { get; set; }
        [DataMember]
        public DateTime? RecuperationExameDate { get; set; }
        [DataMember]
        public DateTime? PracticWorkPresentationDate { get; set; }
        [DataMember]
        public int TotalHoursOfClasses { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Scheduled { get; set; }
    }
}
```

Student.cs

```
using System.Collections.Generic;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Student
    {
        public Student()
        {
            Courses = new List<StudentCourse>();
        }
        [DataMember]
        public IList<StudentCourse> Courses { get; set; }
        [DataMember]
        public Career Career { get; set; }
```

```
        [DataMember]
        public string FirstName { get; set; }
        [DataMember]
        public string LastName { get; set; }
        [DataMember]
        public int StudentCode { get; set; }
        [DataMember]
        public int CareerCode { get; set; }
        [DataMember]
        public string DeviceUri { get; set; }
    }
}
```

StudentCourse.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class StudentCourse
    {
        [DataMember]
        public double PartialExamResult { get; set; }
        [DataMember]
        public double FinalExamResult { get; set; }
        [DataMember]
        public double RecuperationExamResult { get; set; }
        [DataMember]
        public double PracticWorkResult { get; set; }
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public Course Course { get; set; }
        [DataMember]
        public bool FinalExamInscripted { get; set; }

        public double Attendance()
        {
            return (TotalHoursAssisted * 100) / Course.TotalHoursOfClasses;
            //if (result > 75)
            //    return 3;
            //if (result < 75 && result > 50)
            //    return 2;
            //return 3;
        }
    }
}
```

Helpers

AttendanceState.cs

```
namespace TesinaMobileCloud.Data.Helpers
```

```
{
    public enum AttendanceState
    {
        Danger = 1,
        Precaution = 2,
        Good = 3
    }
}
```

JsonSerializationHelper.cs

```
using System.IO;
using System.Runtime.Serialization.Json;
using System.Text;

namespace TesinaMobileCloud.Data.Helpers
{
    public class JsonSerializationHelper
    {
        public static string Serialize<T>(T entity)
        {
            var ser = new DataContractJsonSerializer(typeof(T));
            var ms = new MemoryStream();
            ser.WriteObject(ms, entity);
            var jsonString = Encoding.UTF8.GetString(ms.ToArray());
            ms.Close();
            return jsonString;
        }

        public static T JsonDeserialize<T>(string jsonString)
        {
            var ser = new DataContractJsonSerializer(typeof(T));
            var ms = new MemoryStream(Encoding.UTF8.GetBytes(jsonString));
            T obj = (T)ser.ReadObject(ms);
            return obj;
        }

    }
}
```

Model

AzureEntity.cs

```
using System;
using System.Data.Services.Common;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataServiceEntity]
    [DataContract]
    public class AzureEntity
    {
        public string RowKey { get; set; }
        public string PartitionKey { get; set; }
```

```
        public DateTime Timestamp { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Career : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return RowKey; }
            set { RowKey = value; }
        }

        public Career()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Career(int careerCode, string title)
        {
            CareerCode = careerCode;
            Title = title;
        }


    }
}
```

CareerCourseRelationship.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class CareerCourseRelationship : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
```

```
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public int YearInTheCareer { get; set; }

        public CareerCourseRelationship()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public CareerCourseRelationship(int careerCode, int courseCode)
        {
            CareerCode = careerCode;
            CourseCode = courseCode;
        }
    }
}
```

Course.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Course : AzureEntity
    {
        [DataMember]
        public int Code
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return string.IsNullOrEmpty(RowKey) ? string.Empty : RowKey; }
            set { RowKey = value; }
        }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public DateTime? FinalExamDate { get; set; }
        [DataMember]
        public DateTime? PartialExameDate { get; set; }
        [DataMember]
        public DateTime? PracticWorkPresentationDate { get; set; }
        [DataMember]
```

```
        public DateTime? RecuperationExameDate { get; set; }
        [DataMember]
        public int TotalHoursOfClasses { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Scheduled { get; set; }
         [DataMember]
        public bool IsCritical { get; set; }

        public Course()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Course(int courseCode, string name)
        {
            Code = courseCode;
            Title = name;
        }
    }
}
```

Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Student : AzureEntity
    {
        [DataMember]
        public int StudentCode
        {
            get { return int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CareerCode
        {
            get { return int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public string FirstName { get; set; }
        [DataMember]
        public string LastName { get; set; }
        [DataMember]
```

```
        public string DeviceUri { get; set; }

        public Student()
        {
            PartitionKey = "0";
            RowKey = "0";
        }
    }
}
```

StudentCourseRelationship.cs

```
using System;
using System.Runtime.Serialization;
using TesinaMobileCloud.Data.Helpers;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class StudentCourseRelationship : AzureEntity
    {
        [DataMember]
        public int StudentCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public double PartialExamResult { get; set; }
        [DataMember]
        public double FinalExamResult { get; set; }
        [DataMember]
        public double RecuperationExamResult { get; set; }
        [DataMember]
        public double PracticWorkResult { get; set; }
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public bool FinalExamInscripted { get; set; }

        public StudentCourseRelationship()
        {
            PartitionKey = "0";
            RowKey = "0";
        }

        public AttendanceState GetAttendanceState(int totalHoursOfClasses, bool isCritical)
        {
```

```
                var attendence = (TotalHoursAssisted * 100) / totalHoursOfClasses;
                if (isCritical)
                {
                    if (attendence >= 75)
                        return AttendanceState.Good;
                    if (attendence < 75 && attendence > 50)
                        return AttendanceState.Precaution;
                    return AttendanceState.Danger;
                }
                if (attendence >= 90)
                    return AttendanceState.Good;
                if (attendence < 90 && attendence > 75)
                    return AttendanceState.Precaution;
                return AttendanceState.Danger;
            }
        }
    }
```

Queue

IQueue.cs

```
    namespace TesinaMobileCloud.Data.Queue
    {
        public interface IQueue<T>
        {
            void AddMessage(T message);
            T GetMessage();
            bool HasMessage { get; }
        }


    }
```

NotificationQueue.cs

```
    using System;
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using TesinaMobileCloud.Data.Helpers;

    namespace TesinaMobileCloud.Data.Queue
    {
        public class NotificationQueue<T> : IQueue<T>
        {
            private readonly CloudQueue _queue;
            private const string _queueName = "notificationqueue";

            public NotificationQueue(CloudStorageAccount cloudStorageAccount)
            {
                var cloudQueueClient = cloudStorageAccount.CreateCloudQueueClient();
                _queue = cloudQueueClient.GetQueueReference(_queueName);
                _queue.CreateIfNotExist();
            }
```

```
        public void AddMessage(T message)
        {
            var messageString = JsonSerializationHelper.Serialize<T>(message);
            _queue.AddMessage(new CloudQueueMessage(messageString));
        }

        public T GetMessage()
        {
            var decodedMessage = _queue.GetMessage();
            _queue.DeleteMessage(decodedMessage);
            return JsonSerializationHelper.JsonDeserialize<T>(decodedMessage.AsString);
        }

        public bool HasMessage {
            get
            {
                _queue.FetchAttributes();
                return _queue.ApproximateMessageCount != 0;
            }
        }
    }

    public class PushNotificationErrorsQueue<T> : IQueue<T>
    {

        private readonly CloudQueue _queue;
        private const string QueueName = "pushnotificationerrorsqueue";

        public PushNotificationErrorsQueue(CloudStorageAccount cloudStorageAccount)
        {
            var cloudQueueClient = cloudStorageAccount.CreateCloudQueueClient();
            _queue = cloudQueueClient.GetQueueReference(QueueName);
            _queue.CreateIfNotExist();
        }


        public void AddMessage(T message)
        {
            _queue.AddMessage(new CloudQueueMessage(message.ToString()));
        }

        public T GetMessage()
        {
            throw new System.NotImplementedException();
        }

        public bool HasMessage { get; private set; }
    }
}
```

QueueNotificationMessage.cs

```
using System;
using Microsoft.WindowsAzure.StorageClient;
using TesinaMobileCloud.Data.Helpers;
```

```
namespace TesinaMobileCloud.Data.Queue
{
    public class QueueNotificationMessage
    {
        public string Destinatary { get; set; }

        public string Type { get; set; }

        public string Payload { get; set; }

        public string Priority
        {
            get { return Type.ToLower().Equals("toast") ? "2" : "1"; }
        }
    }
}
```

Repository

CareerCourseRelationshipRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CareerCourseRelationshipRepository :
ICourseCareerRelationshipRepository
    {
        private readonly ITable<CareerCourseRelationship> _table;

        public CareerCourseRelationshipRepository(ITable<CareerCourseRelationship> table)
        {
            _table = table;
        }

        public int GetCareerYearOfCourseByCareer(int careerCode, int courseCode)
        {
            return
                _table.GetSingleByCriteria(
                    relationship => relationship.CareerCode == careerCode &&
relationship.CourseCode == courseCode).
                    YearInTheCareer;
        }
    }
}
```

CareerRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;
```

```
namespace TesinaMobileCloud.Data.Repository
{
    public class CareerRepository : ICareerRepository
    {
        private readonly ITable<Career> _careerTable;

        public CareerRepository(ITable<Career> careerTable)
        {
            _careerTable = careerTable;
        }

        public Career GetCareer(string careerCode)
        {
            return _careerTable.GetSingleByCriteria(career => career.CareerCode ==
int.Parse(careerCode));
        }
    }
}
```

CourseRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CourseRepository : ICourseRepository
    {
        private readonly ITable<Course> _courseTable;
        private readonly ITable<CareerCourseRelationship> _careerCourseTable;

        public CourseRepository(ITable<Course> courseTable,
ITable<CareerCourseRelationship> careerCourseTable)
        {
            _courseTable = courseTable;
            _careerCourseTable = careerCourseTable;
        }

        public IList<Course> GetCoursesByCareer(string careerCode)
        {
            var courses = _careerCourseTable.GetByCriteria(c => c.CareerCode ==
int.Parse(careerCode));
            return courses.Select(careerCourseRelationship =>
_courseTable.GetSingleByCriteria(c => c.Code ==
careerCourseRelationship.CourseCode)).ToList();
        }

        public IList<Course> GetAllCourses()
        {
            return _courseTable.GetByCriteria(c => !string.IsNullOrEmpty(c.PartitionKey));
        }
```

```
        public void AddCourse(Course course)
        {
            _courseTable.Add(course);
        }

        public Course GetSingleCourse(int courseCode)
        {
            return _courseTable.GetSingleByCriteria(c => c.Code == courseCode);
        }

        public void DeleteCourse(int partitionKey)
        {
            _courseTable.Delete(_courseTable.GetSingleByCriteria(c => c.Code ==
partitionKey));
        }
    }
}
```

StudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class StudentCourseRepository : IStudentCourseRepository
    {
        private readonly ITable<StudentCourseRelationship> _studentCourseTable;

        public StudentCourseRepository(ITable<StudentCourseRelationship>
studentCourseTable)
        {
            _studentCourseTable = studentCourseTable;
        }

        public IEnumerable<StudentCourseRelationship> RetriveCoursesOfStudent(string
studentCode)
        {
            return _studentCourseTable.GetByCriteria(s => s.StudentCode ==
int.Parse(studentCode));
        }

        public StudentCourseRelationship RetriveStudentCourseByCode(string studentCode,
string courseCode)
        {
            return _studentCourseTable.GetSingleByCriteria(sc => sc.StudentCode ==
int.Parse(studentCode) && sc.CourseCode == int.Parse(courseCode));
        }
    }
}
```

StudentRepository.cs

```
using TesinaMobileCloud.Data.Model;
```

```
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class StudentRepository : IStudentRepository
    {
        private readonly ITable<Student> _studentTable;

        public StudentRepository(ITable<Student> studentTable)
        {
            _studentTable = studentTable;
        }

        public Student GetStudentByCareerCodeAndStudentCode(string studentCode, string
careerCode)
        {
            return _studentTable.GetSingleByCriteria(s => s.StudentCode ==
int.Parse(studentCode) &&
                                                     s.CareerCode == int.Parse(careerCode));
        }

        public void Add(Student student)
        {
            _studentTable.Add(student);
        }

        public Student GetStudentByStudentCode(string studentCode)
        {
            return _studentTable.GetSingleByCriteria(s => s.StudentCode ==
int.Parse(studentCode));
        }
    }
}
```

ICareerRepository.cs

```
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICareerRepository
    {
        Career GetCareer(string careerCode);
    }
}
```

ICourseCareerRelationshipRepository.cs

```
namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseCareerRelationshipRepository
    {
        int GetCareerYearOfCourseByCareer(int careerCode, int courseCode);
```

```
    }
}
```

ICourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseRepository
    {
        IList<Course> GetCoursesByCareer(string careerCode);
        IList<Course> GetAllCourses();
        void AddCourse(Course course);
        Course GetSingleCourse(int courseCode);
    }
}
```

IStudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface IStudentCourseRepository
    {
        IEnumerable<StudentCourseRelationship> RetriveCoursesOfStudent(string
studentCode);
        StudentCourseRelationship RetriveStudentCourseByCode(string studentCode, string
courseCode);
    }
}
```

IStudentRepository.cs

```
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface IStudentRepository
    {
        Student GetStudentByCareerCodeAndStudentCode(string studentCode, string
careerCode);
        void Add(Student student);
        Student GetStudentByStudentCode(string studentCode);
    }
}
```

Table

AzureTable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;

namespace TesinaMobileCloud.Data.Table
{
   public class AzureTable<T> : ITable<T>
   {
      private readonly TableServiceContext _context;
      private readonly string _tableName;
      private IQueryable<T> Query { get; set; }

      public AzureTable(CloudStorageAccount cloudStorageAccount)
      {
         _tableName = typeof (T).Name;

         var table = new CloudTableClient(cloudStorageAccount.TableEndpoint.ToString(),
cloudStorageAccount.Credentials);
         _context = table.GetDataServiceContext();
         table.CreateTableIfNotExist(_tableName);
         Query = _context.CreateQuery<T>(_tableName).AsTableServiceQuery();
      }

      public IList<T> GetByCriteria(Func<T, bool> func)
      {
         return Query.Where(func).ToList();
      }

      public T GetSingleByCriteria(Func<T, bool> func)
      {
         return Query.SingleOrDefault(func);
      }

      public void Add(T entity)
      {
         try
         {
            _context.AddObject(_tableName, entity);
         }
         catch (InvalidOperationException)
         {
            _context.UpdateObject(entity);
         }
         _context.SaveChanges();

      }

      public void Delete(T entity)
      {
         _context.DeleteObject(entity);
         _context.SaveChanges();
      }
   }
}
```

ITable.cs

```
using System;
using System.Collections.Generic;

namespace TesinaMobileCloud.Data.Table
{
    public interface ITable<T>
    {
        IList<T> GetByCriteria(Func<T, bool> func);
        T GetSingleByCriteria(Func<T, bool> func);
        void Add(T entity);
        void Delete(T entity);
    }
}
```

TesinaMobileCloud.Data.Test

CourseCareerRelationshipRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class CourseCareerRelationshipRepositoryFixture
    {
        private Mock<ITable<CareerCourseRelationship>> _mockCourseCareerRelationship;

        [TestInitialize]
        public void Setup()
        {
            _mockCourseCareerRelationship = new
Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCareerYearOfCourseByCareer()
        {
            var career = 502;
            var course = 120;
            _mockCourseCareerRelationship.Setup(
                m => m.GetSingleByCriteria(It.IsAny<Func<CareerCourseRelationship,
bool>>())).Returns(new CareerCourseRelationship
                                                                        {
                                                                            CareerCode = career,
                                                                            CourseCode =
course,
                                                                            YearInTheCareer = 1
                                                                        });
```

```
            var sut = new
        CareerCourseRelationshipRepository(_mockCourseCareerRelationship.Object);
            var result = sut.GetCareerYearOfCourseByCareer(career, course);

            Assert.AreEqual(1, result);


        }
    }
}
```

CourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class CourseRepositoryFixture
    {
        private Mock<ITable<Course>> _mockCourse;
        private Mock<ITable<CareerCourseRelationship>> _mockCareerCourse;

        [TestInitialize]
        public void Setup()
        {
            _mockCourse = new Mock<ITable<Course>>();
            _mockCareerCourse = new Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCoursesByCareer()
        {
            const int careerCode = 502;
            _mockCareerCourse.Setup(m =>
        m.GetByCriteria(It.IsAny<Func<CareerCourseRelationship, bool>>())).Returns(new
        List<CareerCourseRelationship>
                                                                    {
                                                                        new
        CareerCourseRelationship(502, 123),
                                                                        new
        CareerCourseRelationship(502, 124),
                                                                    });
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
        bool>>())).Returns(new Course(123, "Programación II"));
            var sut = new CourseRepository(_mockCourse.Object,
        _mockCareerCourse.Object);
```

```
            var result = sut.GetCoursesByCareer(careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 2);
        }

        [TestMethod]
        public void GetAllCourses()
        {
            _mockCourse.Setup(m => m.GetByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new List<Course>{ new Course(123, "Programación II")});
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 1);
        }

        [TestMethod]
        public void DeleteCourse()
        {
            var course = new Course(122, "Programación I");
            var courses = new List<Course>
                        {
                            new Course(123, "Programación II"),
                            course
                        };
            _mockCourse.Setup(m => m.Delete(course)).Callback(() =>
courses.Remove(course));
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(course);

            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            sut.DeleteCourse(122);

            Assert.AreEqual(1, courses.Count);
        }

        [TestMethod]
        public void AddCourse()
        {
            var course = new Course(123, "Programación II");
            _mockCourse.Setup(m => m.Add(It.IsAny<Course>())).Verifiable();

            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);
            sut.AddCourse(course);
            _mockCourse.Verify();
        }
    }
```

```
}
```

StudentCourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
   [TestClass]
   public class StudentCourseRepositoryFixture
   {
      private Mock<ITable<StudentCourseRelationship>> _studentCourseTable;

      [TestInitialize]
      public void Setup()
      {
         _studentCourseTable = new Mock<ITable<StudentCourseRelationship>>();
      }
      [TestMethod]
      public void RetriveCoursesOfStudent()
      {
         _studentCourseTable.Setup(m =>
m.GetByCriteria(It.IsAny<Func<StudentCourseRelationship, bool>>()))
                        .Returns(new List<StudentCourseRelationship>());
         var sut = new StudentCourseRepository(_studentCourseTable.Object);

         var result = sut.RetriveCoursesOfStudent("10004");

         Assert.IsNotNull(result);
         Assert.IsInstanceOfType(result,
typeof(IEnumerable<StudentCourseRelationship>));
      }

      [TestMethod]
      public void RetriveStudentCourseByCode()
      {
         const string studentCode = "2";
         const string courseCode = "3";
         _studentCourseTable.Setup(m =>
m.GetSingleByCriteria(It.IsAny<Func<StudentCourseRelationship, bool>>()))
                        .Returns(new StudentCourseRelationship
                           {
                              StudentCode = int.Parse(studentCode),
                              CourseCode = int.Parse(courseCode)
                           });
         var sut = new StudentCourseRepository(_studentCourseTable.Object);

         var result = sut.RetriveStudentCourseByCode(studentCode, courseCode);
```

```
            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof (StudentCourseRelationship));
            Assert.AreEqual(studentCode, result.StudentCode);
            Assert.AreEqual(courseCode, result.CourseCode);
        }
    }
}
```

StudentRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class StudentRepositoryFixture
    {
        private Mock<ITable<Student>> _mockStudentTable;

        [TestInitialize]
        public void Setup()
        {
            _mockStudentTable = new Mock<ITable<Student>>();
        }

        [TestMethod]
        public void GetStudentByCareerCodeAndStudentCode()
        {
            const int studentCode = 10004;
            const int careerCode = 502;
            _mockStudentTable.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Student,
bool>>()))
                             .Returns(new Student
                                      {
                                          CareerCode = careerCode,
                                          StudentCode = studentCode
                                      });

            var sut = new StudentRepository(_mockStudentTable.Object);

            var result = sut.GetStudentByCareerCodeAndStudentCode(studentCode.ToString(),
careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(10004, result.StudentCode);
            Assert.AreEqual(502, result.CareerCode);
        }
    }
}
```

UnitTest1.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class QueueTest
    {
        [TestMethod]
        public void TestMethod1()
        {
        }
    }
}
```

TesinaMobileCloud.Notifications

IPushNotification.cs

```
namespace TesinaMobileCloud.Notifications
{
    public interface IPushNotification
    {
        string Type { get; }
        string Payload { get; }
    }
}
```

TileNotification.cs

```
using System;
namespace TesinaMobileCloud.Notifications
{
    public class TileNotification : IPushNotification
    {
        public string BackTitle { get; set; }

        public string BackContent { get; set; }

        private string _backBackgroundImage;

        public string BackBackgroundImage
        {
            get { return _backBackgroundImage; }
            set { _backBackgroundImage = new
Uri(String.Format("/BackgroundImages/{0}.png", value), UriKind.Relative).ToString(); }
        }

        public string Payload
        {
            get
            {
```

```
                var tileMessage = string.Format("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                                "<wp:Notification xmlns:wp=\"WPNotification\">" +
                                                   "<wp:Tile >" +
                                                      "<wp:BackgroundImage>" +
                                                      "</wp:BackgroundImage>" +
                                                      "<wp:Count>" +
                                                      "</wp:Count>" +
                                                      "<wp:Title>" +
                                                      "</wp:Title>" +

            "<wp:BackBackgroundImage>{0}</wp:BackBackgroundImage>" +
                                                      "<wp:BackTitle>{1}</wp:BackTitle>" +
                                                      "<wp:BackContent>{2}</wp:BackContent>" +
                                                   "</wp:Tile> " +
                                                "</wp:Notification>", BackBackgroundImage, BackTitle,
            BackContent);

                    return tileMessage;
                }
            }


            public string Type
            {
                get { return "token"; }
            }
        }
    }
```

ToastNotification.cs

```
    namespace TesinaMobileCloud.Notifications
    {
        public class ToastNotification : IPushNotification
        {
            public string Title { get; set; }

            public string Content { get; set; }

            public string Type
            {
                get { return "toast"; }
            }

            public string Payload
            {
                get
                {
                    return string.Format("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                         "<wp:Notification xmlns:wp=\"WPNotification\">" +
                                            "<wp:Toast>" +
                                               "<wp:Text1>{0}</wp:Text1>" +
                                               "<wp:Text2>{1}</wp:Text2>" +
                                               "<wp:Param>{2}</wp:Param>" +
                                            "</wp:Toast>" +
```

```
                                "</wp:Notification>", Title, Content, Page);
            }
        }

        public string Page { get; set; }
    }
}
```

TesinaMobileCloud.Phone.Agents

ScheduledAgent.cs

```
using System;
using System.Windows;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Data.Repositories;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Phone.Services.Support;
using Microsoft.Phone.Shell;
using System.Linq;

namespace TesinaMobileCloud.Phone.Agents
{
    public class ScheduledAgent : ScheduledTaskAgent
    {
        private static volatile bool _classInitialized;
        private readonly ICloudService _cloudService;
        private IStudentInformationManager _studentInformationManager;

        /// <remarks>
        /// ScheduledAgent constructor, initializes the UnhandledException handler
        /// </remarks>
        public ScheduledAgent()
        {
            if (!_classInitialized)
            {
                _classInitialized = true;
                // Subscribe to the managed exception handler
                Deployment.Current.Dispatcher.BeginInvoke(delegate
                {
                    Application.Current.UnhandledException +=
ScheduledAgent_UnhandledException;
                });
            }

            _cloudService = new CloudService();
            _studentInformationManager = new StudentInformationManager(new
StudentRepository(),
                                        new StudentCourseRepository(),
                                        new ScheduleActionServiceAdapter());
```

```
        }

        /// Code to execute on Unhandled Exceptions
        private void ScheduledAgent_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        /// <summary>
        /// Agent that runs a scheduled task
        /// </summary>
        /// <param name="task">
        /// The invoked task
        /// </param>
        /// <remarks>
        /// This method is called when a periodic or resource intensive task is invoked
        /// </remarks>
        protected override void OnInvoke(ScheduledTask task)
        {
            if (task is PeriodicTask)
            {
                PerformSyncronizationTask();
            }
        }
        private void PerformSyncronizationTask()
        {
            var syncService = new SyncronizationService(_cloudService,
_studentInformationManager);

            syncService.SyncCourses()
                    .ObserveOnDispatcher()
                    .Subscribe(SyncCompleted, SyncFailed);
        }

        private void SyncCompleted(TaskSummary result)
        {
            //DisplayShellNotification(new StandardTileData
            //                        {
            //                            Title = "SIA",
            //                            BackContent = "Información Actualizada"
            //                        });
            NotifyComplete();
        }

        private void SyncFailed(Exception exception)
        {
            //DisplayShellNotification(new StandardTileData
            //                        {
            //                            Title = "SIA",
```

```
                //                      BackContent = "Error al sincronizar datos"
                //                  });
                Abort();
            }

            private static void DisplayShellNotification(ShellTileData tileData)
            {
                var tile = ShellTile.ActiveTiles.First();
                tile.Update(tileData);
            }
        }
    }
```

TesinaMobileCloud.Phone.Client

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone" xmlns:sys="clr-
namespace:System;assembly=mscorlib"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.ViewModel" mc:Ignorable="d">
    <!--Application Resources-->
    <Application.Resources>
        <ResourceDictionary>
            <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
            <Style TargetType="TextBlock" x:Key="TextBlockHeader">
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="32" />
                <Setter Property="TextWrapping" Value="Wrap"/>
                <Setter Property="FontFamily" Value="Segoe WP Semibold"/>
                <Setter Property="Margin" Value="0,0,0,5"/>
            </Style>
            <Style TargetType="TextBlock" x:Key="TextBlockContent">
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="25" />
                <Setter Property="TextWrapping" Value="Wrap"/>
            </Style>
            <SolidColorBrush x:Key="PhoneControlBackgroundBrush" Color="#FF0083B6"/>
            <Color x:Key="ApplicationBarBrush">#FF0F5CE5</Color>
        </ResourceDictionary>
    </Application.Resources>
    <Application.ApplicationLifetimeObjects>
        <!--Required object that handles lifetime events for the application-->
        <shell:PhoneApplicationService Launching="Application_Launching"
                                       Closing="Application_Closing"
                                       Activated="Application_Activated"
                                       Deactivated="Application_Deactivated" />
    </Application.ApplicationLifetimeObjects>
</Application>
```

App.xaml.cs

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        protected ViewModelLocator Locator
        {
            get { return (ViewModelLocator)Resources["Locator"]; }
        }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            TiltEffect.TiltableItems.Add(typeof(CourseResumeViewModel));
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;
```

```
            // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
            // application's PhoneApplicationService object to Disabled.
            // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
            // and consume battery power when the user is not using the phone.
            PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
        }

    }

    // Code to execute when the application is launching (eg, from Start)
    // This code will not execute when the application is reactivated
    private void Application_Launching(object sender, LaunchingEventArgs e)
    {
        Locator.ScheduleActions.RemovePeriodicTask();
    }

    // Code to execute when the application is activated (brought to foreground)
    // This code will not execute when the application is first launched
    private void Application_Activated(object sender, ActivatedEventArgs e)
    {

    }

    // Code to execute when the application is deactivated (sent to background)
    // This code will not execute when the application is closing
    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
    {
        // Ensure that required application state is persisted here.
    }

    // Code to execute when the application is closing (eg, user hit Back)
    // This code will not execute when the application is deactivated
    private void Application_Closing(object sender, ClosingEventArgs e)
    {
        Locator.ScheduleActions.AddPeriodicTask();
    }

    // Code to execute if a navigation fails
    private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
    {
        if (System.Diagnostics.Debugger.IsAttached)
        {
            // A navigation has failed; break into the debugger
            System.Diagnostics.Debugger.Break();
        }
    }

    // Code to execute on Unhandled Exceptions
    private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
```

```
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new TransitionFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
    e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

MainPage.xaml

```
<client:PhonePage
    x:Class="TesinaMobileCloud.Phone.Client.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
```

```
    xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:View="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:client="clr-namespace:TesinaMobileCloud.Phone.Client"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="800"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"
    Orientation="Portrait"
    DataContext="{Binding Main, Source={StaticResource Locator}}"
    shell:SystemTray.IsVisible="False">
    <toolkit:TransitionService.NavigationInTransition>
        <toolkit:NavigationInTransition>
            <toolkit:NavigationInTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardIn"/>
            </toolkit:NavigationInTransition.Backward>
            <toolkit:NavigationInTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardIn"/>
            </toolkit:NavigationInTransition.Forward>
        </toolkit:NavigationInTransition>
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
        <toolkit:NavigationOutTransition>
            <toolkit:NavigationOutTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardOut"/>
            </toolkit:NavigationOutTransition.Backward>
            <toolkit:NavigationOutTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardOut"/>
            </toolkit:NavigationOutTransition.Forward>
        </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

            <client:PhonePage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Mode="Minimized"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="0.35">
            <shell:ApplicationBar.MenuItems>
                        <shell:ApplicationBarMenuItem Text="Configuración" IsEnabled="True"
Click="NavigateToSettingsView"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </client:PhonePage.ApplicationBar>

            <Grid x:Name="LayoutRoot" Background="Transparent">
        <controls:Panorama Title="Sia" Background="{StaticResource
PhoneControlBackgroundBrush}">
            <controls:PanoramaItem Header="Cátedras">
                <View:CoursesView/>
            </controls:PanoramaItem>
        </controls:Panorama>
```

```
    </Grid>
</client:PhonePage>
```

MainPage.xaml.cs

```
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client
{
    public partial class MainPage : PhonePage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        private void NavigateToSettingsView(object sender, System.EventArgs e)
        {
            var viewModel = DataContext as MainViewModel;

            if (viewModel != null)
                viewModel.NavigateToSettingsView.Execute(null);
        }
    }
}
```

PhonePage.cs

```
using System.Windows.Navigation;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client
{
    public class PhonePage : PhoneApplicationPage
    {
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            var dataContext = DataContext as ViewModel.ViewModel;
            if (e.NavigationMode == NavigationMode.Back) return;

            if (dataContext != null)
                dataContext.OnNavigateTo(NavigationContext.QueryString);

            base.OnNavigatedTo(e);
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            var dataContext = DataContext as ViewModel.ViewModel;
            if (e.NavigationMode == NavigationMode.Forward) return;

            if (dataContext != null)
                dataContext.OnNavigateFrom(NavigationContext.QueryString);

            base.OnNavigatedFrom(e);
```

```
        }
    }
}
```

View

CourseDescriptionView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client"
x:Class="TesinaMobileCloud.Phone.Client.CourseDescriptionView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    DataContext="{Binding CourseDescription, Source={StaticResource Locator}}"
    d:DesignHeight="607" d:DesignWidth="456">

    <Grid x:Name="LayoutRoot">
        <ListBox HorizontalAlignment="Left" Height="607" VerticalAlignment="Top"
Width="456">
            <TextBlock x:Name="Description" Margin="0,0,0,10" Style="{StaticResource
TextBlockContent}" Text="{Binding Description}" Width="456"/>
            <TextBlock x:Name="Professor" Style="{StaticResource TextBlockHeader}"
Text="Docente"/>
            <TextBlock x:Name="ProfessorName" Text="{Binding Professor}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="DayAndTime" Style="{StaticResource TextBlockHeader}"
Text="Horario"/>
            <TextBlock x:Name="DayAndTimeContent" Text="{Binding Scheduled}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PartialExamDate" Style="{StaticResource TextBlockHeader}"
Text="Examen Parcial"/>
            <TextBlock x:Name="PartialExamDateContent" Text="{Binding PartialExamDate}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="RecuperationExameDate" Text="Examen Recuperatorio"
Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="RecuperationExameDateContent" Text="{Binding
RecuperationExamDate}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="FinalExamDate" Text="Examen Final" Style="{StaticResource
TextBlockHeader}"/>
            <TextBlock x:Name="FinalExamDateContent" Text="{Binding FinalExamDate}"
Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PracticWorkExameDate" Text="Trabajos Practicos"
Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="PracticWorkExameDateContent" Text="{Binding
PracticWorkExamDate}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="TotalHoursOfClasses" Text="Total de horas de cursada"
Style="{StaticResource TextBlockHeader}"/>
```

```
            <TextBlock x:Name="TotalHoursOfClassesContent" Text="{Binding
        TotalHoursOfClasses}" Style="{StaticResource TextBlockContent}"/>
            </ListBox>
        </Grid>
    </UserControl>
```

CourseDescriptionView.xaml.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Windows;
    using System.Windows.Controls;
    using System.Windows.Navigation;
    using Microsoft.Phone.Controls;
    using Microsoft.Phone.Shell;

    namespace TesinaMobileCloud.Phone.Client
    {
        public partial class CourseDescriptionView : UserControl
        {
            public CourseDescriptionView()
            {
                InitializeComponent();
            }
        }
    }
```

CourseResumeView.xaml

```
    <UserControl
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
            xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
            xmlns:i="clr-
    namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
    xmlns:ec="clr-
    namespace:Microsoft.Expression.Interactivity.Core;assembly=Microsoft.Expression.Interact
    ions" xmlns:Command="clr-
    namespace:GalaSoft.MvvmLight.Command;assembly=GalaSoft.MvvmLight.Extras.WP71"
    x:Class="TesinaMobileCloud.Phone.Client.View.CourseResumeView"
            mc:Ignorable="d"
            FontFamily="{StaticResource PhoneFontFamilyNormal}"
            FontSize="{StaticResource PhoneFontSizeNormal}"
            Foreground="{StaticResource PhoneForegroundBrush}"
            d:DesignHeight="85" d:DesignWidth="432" RenderTransformOrigin="0.5,0.5">

            <Grid x:Name="LayoutRoot" Margin="0,0,0,10">
                    <Grid Height="75" VerticalAlignment="Top">
            <i:Interaction.Triggers>
                <i:EventTrigger EventName="Tap">
                    <Command:EventToCommand Command="{Binding CoursePageDetails}"/>
                </i:EventTrigger>
```

```
        </i:Interaction.Triggers>
        <TextBlock x:Name="Title" HorizontalAlignment="Left" Margin="25,-9,0,0"
VerticalAlignment="Top" Height="52" FontSize="37" Text="{Binding Title}"/>
                        <TextBlock x:Name="Professor" HorizontalAlignment="Left"
Margin="25,48,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Text="{Binding
Professor}"/>
                        <Border x:Name="Attendance" Background="{Binding
AttendanceState}" HorizontalAlignment="Left" Height="75" VerticalAlignment="Top"
Width="20"/>
                    </Grid>
        </Grid>
</UserControl>
```

CourseResumeView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CourseResumeView : UserControl
    {
        public CourseResumeView()
        {
            InitializeComponent();
        }
    }
}
```

CourseStudentSituationView.xaml

```
<UserControl x:Class="TesinaMobileCloud.Phone.Client.CourseStudentSituationView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="607" d:DesignWidth="456">

    <Grid x:Name="LayoutRoot">
        <ListBox HorizontalAlignment="Left" Height="607" VerticalAlignment="Top"
Width="456">
        <TextBlock x:Name="PartialExamResult" Text="Resultado Examen Parcial"
Style="{StaticResource TextBlockHeader}" Margin="0"/>
```

```
            <TextBlock x:Name="PartialExamResultContent" Text="{Binding
        PartialExamResult}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="RecuperationExameResult" Style="{StaticResource
        TextBlockHeader}">
                                <Run Text="Calificación "/>
                                <Run Text="Examen Recup"/>
                                <Run Text="."/>
                        </TextBlock>
            <TextBlock x:Name="RecuperationExameResultContent" Text="{Binding
        RecuperationExamResult}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="FinalExamResult" Text="Resultado Examen Final"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="FinalExamResultContent" Text="{Binding FinalExamResult}"
        Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PracticWorkExameResult" Style="{StaticResource
        TextBlockHeader}">
                                <Run Text="Resultado"/>
                                <Run Text=" "/>
                                <Run Text="de Trab"/>
                                <Run Text="."/>
                                <Run Text=" Practicos"/>
                        </TextBlock>
            <TextBlock x:Name="PracticWorkExameResultContent" Text="{Binding
        PracticWorkExamResult}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="AssistedHoursOfClasses" Text="Total de Horas Asistidas"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="AssistedHoursOfClassesContent" Text="{Binding
        TotalAssistedHours}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="AssistedPercent" Text="Porcentaje de Asistencias"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="AssistedPercentContent" Text="{Binding
        ActualAssistedPercent}" Style="{StaticResource TextBlockContent}"/>
             </ListBox>

       </Grid>
    </UserControl>
```

CourseStudentSituationView.xaml.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Windows;
    using System.Windows.Controls;
    using System.Windows.Navigation;
    using Microsoft.Phone.Controls;
    using Microsoft.Phone.Shell;

    namespace TesinaMobileCloud.Phone.Client
    {
       public partial class CourseStudentSituationView : UserControl
       {
          public CourseStudentSituationView()
          {
```

```
            InitializeComponent();
        }
    }
}
```

CoursesView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    x:Class="TesinaMobileCloud.Phone.Client.View.CoursesView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="498" d:DesignWidth="420"
    DataContext="{Binding Courses, Source={StaticResource Locator}}">
    <Grid x:Name="Courses">
        <toolkit:LongListSelector x:Name="Selector" ItemsSource="{Binding Courses}"
Background="{x:Null}" IsFlatList="True">
            <toolkit:LongListSelector.GroupItemTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding GroupName}" FontSize="32" Foreground="Green"
/>
                </DataTemplate>
            </toolkit:LongListSelector.GroupItemTemplate>
            <toolkit:LongListSelector.ItemTemplate>
                <DataTemplate>
                        <local:CourseResumeView Height="85" Width="432"
toolkit:TiltEffect.IsTiltEnabled="True"/>
                </DataTemplate>
            </toolkit:LongListSelector.ItemTemplate>
        </toolkit:LongListSelector>
    </Grid>
</UserControl>
```

CoursesView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
```

```
namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CoursesView : UserControl
    {
        public CoursesView()
        {
            InitializeComponent();
        }
    }
}
```

CourseView.xaml

```
<Client:PhonePage xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                  xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
                  xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
                  xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
                  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                  xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                  xmlns:Client="clr-namespace:TesinaMobileCloud.Phone.Client"
                  xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:ec="clr-
namespace:Microsoft.Expression.Interactivity.Core;assembly=Microsoft.Expression.Interact
ions"
    x:Class="TesinaMobileCloud.Phone.Client.View.CourseView"
    d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    DataContext="{Binding Course, Source={StaticResource Locator}}"
    SupportedOrientations="Portrait"
    Orientation="Portrait"
    mc:Ignorable="d"
    Opacity="1"
    Foreground="{StaticResource PhoneControlBackgroundBrush}"
    Background="{StaticResource PhoneControlBackgroundBrush}"
    BorderBrush="{StaticResource PhoneControlBackgroundBrush}"
    shell:SystemTray.IsVisible="True"
          shell:SystemTray.BackgroundColor="{StaticResource ApplicationBarBrush}"
    shell:SystemTray.Opacity="0">
    <toolkit:TransitionService.NavigationInTransition>
        <toolkit:NavigationInTransition>
            <toolkit:NavigationInTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardIn"/>
            </toolkit:NavigationInTransition.Backward>
            <toolkit:NavigationInTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardIn"/>
            </toolkit:NavigationInTransition.Forward>
        </toolkit:NavigationInTransition>
```

```
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
        <toolkit:NavigationOutTransition>
            <toolkit:NavigationOutTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardOut"/>
            </toolkit:NavigationOutTransition.Backward>
            <toolkit:NavigationOutTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardOut"/>
            </toolkit:NavigationOutTransition.Forward>
        </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

    <Client:PhonePage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="1">
            <shell:ApplicationBar.MenuItems>
                    <shell:ApplicationBarMenuItem Text="Inscribirse a examen final"
Click="ApplicationBarIconButton_Click_1" IsEnabled="True"/>
            </shell:ApplicationBar.MenuItems>

                        <shell:ApplicationBarIconButton
IconUri="/Assets/AppBar/appbar.refresh.rest.png" Text="Actualizar"
Click="ApplicationBarIconButton_Click_1"/>
        </shell:ApplicationBar>
    </Client:PhonePage.ApplicationBar>
    <!--LayoutRoot is the root grid where all page content is placed-->
    <shell:SystemTray.ProgressIndicator>
        <shell:ProgressIndicator Text="Actualizando" IsVisible="{Binding Updating}"
IsIndeterminate="True" />
    </shell:SystemTray.ProgressIndicator>
    <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
        <!--Pivot Control-->
        <!--<ProgressBar Background="{StaticResource PhoneControlBackgroundBrush}"
Foreground="#FF033A9B" VerticalAlignment="Top" IsIndeterminate="True"/>-->
        <!--<ProgressBar x:Name="ProgressBar" Background="{StaticResource
PhoneControlBackgroundBrush}" Foreground="#FF033A9B" IsIndeterminate="True"
VerticalAlignment="Top"/>-->
        <controls:Pivot Title="{Binding Title}" SelectedIndex="1" Margin="0,15,0,0"
FontSize="26">
            <controls:PivotItem Header="Descripción">
                <Client:CourseDescriptionView DataContext="{Binding
CourseDescriptionViewModel}"/>
            </controls:PivotItem>
            <!--Pivot item two-->
            <controls:PivotItem Header="Mi Situación">
                <Client:CourseStudentSituationView DataContext="{Binding
CourseStudentSituationViewModel}"/>
            </controls:PivotItem>
        </controls:Pivot>
    </Grid>
</Client:PhonePage>
```

CourseView.xaml.cs

```
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CourseView : PhonePage
    {
        public CourseView()
        {
            InitializeComponent();
        }

        private void ApplicationBarIconButton_Click_1(object sender, System.EventArgs e)
        {
            var dataContext = DataContext as CourseViewModel;
            if (dataContext != null)
            {
                dataContext.SyncCourseInfo.Execute(null);
            }
        }
    }
}
```

SettingsView.xaml

```
<client:PhonePage xmlns:Primitives="clr-
namespace:Microsoft.Phone.Controls.Primitives;assembly=Microsoft.Phone.Controls.Toolki
t"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
                xmlns:client="clr-namespace:TesinaMobileCloud.Phone.Client"
                x:Class="TesinaMobileCloud.Phone.Client.View.SettingsView"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d"
    DataContext="{Binding Settings, Source={StaticResource Locator}}"
    shell:SystemTray.IsVisible="True"
          shell:SystemTray.BackgroundColor="{StaticResource ApplicationBarBrush}"
    shell:SystemTray.Opacity="0">
          <toolkit:TransitionService.NavigationInTransition>
       <toolkit:NavigationInTransition>
          <toolkit:NavigationInTransition.Backward>
             <toolkit:TurnstileTransition Mode="BackwardIn"/>
          </toolkit:NavigationInTransition.Backward>
          <toolkit:NavigationInTransition.Forward>
             <toolkit:TurnstileTransition Mode="ForwardIn"/>
          </toolkit:NavigationInTransition.Forward>
```

```
            </toolkit:NavigationInTransition>
        </toolkit:TransitionService.NavigationInTransition>
        <toolkit:TransitionService.NavigationOutTransition>
            <toolkit:NavigationOutTransition>
                <toolkit:NavigationOutTransition.Backward>
                    <toolkit:TurnstileTransition Mode="BackwardOut"/>
                </toolkit:NavigationOutTransition.Backward>
                <toolkit:NavigationOutTransition.Forward>
                    <toolkit:TurnstileTransition Mode="ForwardOut"/>
                </toolkit:NavigationOutTransition.Forward>
            </toolkit:NavigationOutTransition>
        </toolkit:TransitionService.NavigationOutTransition>

        <client:PhonePage.ApplicationBar>
            <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="1">
                <shell:ApplicationBarIconButton
IconUri="/Toolkit.Content/ApplicationBar.Check.png" Text="Guardar"
Click="SaveChangesInSettings"/>
            </shell:ApplicationBar>
        </client:PhonePage.ApplicationBar>
        <shell:SystemTray.ProgressIndicator>
            <shell:ProgressIndicator Text="Actualizando" IsVisible="{Binding Updating}"
IsIndeterminate="True" />
        </shell:SystemTray.ProgressIndicator>
        <!--LayoutRoot is the root grid where all page content is placed-->
        <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

            <!--TitlePanel contains the name of the application and page title-->
            <StackPanel Grid.Row="0" Margin="12,17,0,28">
                <TextBlock Text="Configuración" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}" FontSize="65"/>
            </StackPanel>

            <!--ContentPanel - place additional content here-->
            <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
                <toolkit:ToggleSwitch HorizontalAlignment="Left" VerticalAlignment="Top"
Margin="0,58,0,0" Width="444" IsChecked="{Binding IsChecked, Mode=TwoWay}"/>
                <TextBlock HorizontalAlignment="Left" Margin="0,10,0,0" TextWrapping="Wrap"
Text="Recibir Notificaciones" VerticalAlignment="Top" FontSize="32"/>
            </Grid>
        </Grid>

    </client:PhonePage>
```

SettingsView.xaml.cs

```
using Microsoft.Phone.Controls;
using TesinaMobileCloud.Phone.Client.ViewModel;
```

```
namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class SettingsView : PhonePage
    {
        public SettingsView()
        {
            InitializeComponent();
        }

        private void SaveChangesInSettings(object sender, System.EventArgs e)
        {
            var dataContext = DataContext as SettingsViewModel;
            if (dataContext != null)
                dataContext.SaveChangesInSettings.Execute(null);
        }
    }
}
```

ViewModel

CourseDescriptionViewModel.cs

```
using System;
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseDescriptionViewModel : ViewModel
    {
        private string _textToDisplayIfDateTimeIsNotValid = "A Confirmar...";

        /// <summary>
        /// The <see cref="Description" /> property's name.
        /// </summary>
        public const string DescriptionPropertyName = "Description";

        private string _description = string.Empty;

        /// <summary>
        /// Sets and gets the Description property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Description
        {
            get
            {
                return _description;
            }
            set
            {
                Set(() => Description, ref _description, value);
            }
        }
```

```
/// <summary>
/// The <see cref="Professor" /> property's name.
/// </summary>
public const string ProfessorPropertyName = "Professor";

private string _professor = string.Empty;

/// <summary>
/// Sets and gets the Professor property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Professor
{
    get
    {
        return _professor;
    }
    set
    {
        Set(() => Professor, ref _professor, value);
    }
}

/// <summary>
/// The <see cref="Scheduled" /> property's name.
/// </summary>
public const string ScheduledPropertyName = "Scheduled";

private string _scheduled = string.Empty;

/// <summary>
/// Sets and gets the Scheduled property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Scheduled
{
    get
    {
        return _scheduled;
    }

    set
    {
        if (_scheduled == value)
        {
            return;
        }

        RaisePropertyChanging(() => Scheduled);
        _scheduled = value;
        RaisePropertyChanged(() => Scheduled);
    }
}
```

```
        /// <summary>
        /// The <see cref="PartialExamDate" /> property's name.
        /// </summary>
        public const string PartialExamDatePropertyName = "PartialExamDate";

        private string _partial = string.Empty;

        /// <summary>
        /// Sets and gets the PartialExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string PartialExamDate
        {
            get
            {
                return _partial;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => PartialExamDate, ref _partial, _textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PartialExamDate, ref _partial, value);
            }
        }

        /// <summary>
        /// The <see cref="RecuperationExamDate" /> property's name.
        /// </summary>
        public const string RecuperationExamDatePropertyName = "RecuperationExamDate";

        private string _recuperationExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the RecuperationExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string RecuperationExamDate
        {
            get
            {
                return _recuperationExamDate;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => RecuperationExamDate, ref _recuperationExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => RecuperationExamDate, ref _recuperationExamDate, value);
```

```
            }
        }

        /// <summary>
        /// The <see cref="FinalExamDate" /> property's name.
        /// </summary>
        public const string FinalExamDatePropertyName = "FinalExamDate";

        private string _finalExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the FinalExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string FinalExamDate
        {
            get
            {
                return _finalExamDate;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => FinalExamDate, ref _finalExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => FinalExamDate, ref _finalExamDate, value);
            }
        }

        /// <summary>
        /// The <see cref="PracticWorkExamDate" /> property's name.
        /// </summary>
        public const string PracticWorkExamDatePropertyName = "PracticWorkExamDate";

        private string _practicWorkExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the PracticWorkExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string PracticWorkExamDate
        {
            get
            {
                return _practicWorkExamDate;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
```

```
                Set(() => PracticWorkExamDate, ref _practicWorkExamDate,
_textToDisplayIfDateTimeIsNotValid);
            }
            else
                Set(() => PracticWorkExamDate, ref _practicWorkExamDate, value);
        }
    }

    /// <summary>
    /// The <see cref="TotalHoursOfClasses" /> property's name.
    /// </summary>
    public const string TotalHoursOfClassesPropertyName = "TotalHoursOfClasses";

    private int _totalHoursOfClasses;

    /// <summary>
    /// Sets and gets the TotalHoursOfClasses property.
    /// Changes to that property's value raise the PropertyChanged event.
    /// </summary>
    public int TotalHoursOfClasses
    {
        get
        {
            return _totalHoursOfClasses;
        }
        set
        {
            Set(() => TotalHoursOfClasses, ref _totalHoursOfClasses, value);
        }
    }

  }
}
```

CourseResumeViewModel.cs

```
using System;
using System.Windows.Media;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseResumeViewModel : ViewModel
    {
        public CourseResumeViewModel()
        {
        }

        /// <summary>
        /// The <see cref="Title" /> property's name.
        /// </summary>
        public const string TitlePropertyName = "Title";

        private string _title = string.Empty;
```

```
/// <summary>
/// Sets and gets the Title property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Title
{
   get
   {
      return _title;
   }
   set
   {
      Set(TitlePropertyName, ref _title, value);
   }
}

/// <summary>
/// The <see cref="Professor" /> property's name.
/// </summary>
public const string ProfessorPropertyName = "Professor";

private string _professor = string.Empty;

/// <summary>
/// Sets and gets the Professor property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Professor
{
   get
   {
      return _professor;
   }
   set
   {
      Set(ProfessorPropertyName, ref _professor, value);
   }
}
/// <summary>
/// The <see cref="Attendance" /> property's name.
/// </summary>
public const string AttendancePropertyName = "Attendance";

private double _attendance = 0;

/// <summary>
/// Sets and gets the Attendance property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public double Attendance
{
   get
   {
```

```
        return _attendance;
    }
    set
    {
        Set(AttendancePropertyName, ref _attendance, value);
    }
}

public SolidColorBrush AttendanceState
{
    get
    {
        if (Attendance >= 75)
            return new SolidColorBrush(Colors.Green);
        if (Attendance < 75 && Attendance > 50)
            return new SolidColorBrush(Colors.Yellow);
        return new SolidColorBrush(Colors.Red);
    }
}

/// <summary>
/// The <see cref="Code" /> property's name.
/// </summary>
public const string CodePropertyName = "Code";

private int _code = 0;

/// <summary>
/// Sets and gets the Code property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int Code
{
    get
    {
        return _code;
    }
    set
    {
        Set(CodePropertyName, ref _code, value);
    }
}

/// <summary>
/// The <see cref="StudentCode" /> property's name.
/// </summary>
public const string StudentCodePropertyName = "StudentCode";

private int _studentCode = 0;

/// <summary>
/// Sets and gets the StudentCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
```

```
        public int StudentCode
        {
            get
            {
                return _studentCode;
            }
            set
            {
                Set(() => StudentCode, ref _studentCode, value);
            }
        }

        /// <summary>
        /// The <see cref="CareerCode" /> property's name.
        /// </summary>
        public const string CareerCodePropertyName = "CareerCode";

        private int _careerCode = 0;

        /// <summary>
        /// Sets and gets the CareerCode property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public int CareerCode
        {
            get
            {
                return _careerCode;
            }
            set
            {
                Set(() => CareerCode, ref _careerCode, value);
            }
        }

        private RelayCommand _coursePageDetails;
        public RelayCommand CoursePageDetails
        {
            get
            {
                return _coursePageDetails
                    ?? (_coursePageDetails = new RelayCommand(() =>
NavigationService.NavigateTo(new
Uri(String.Format("/View/CourseView.xaml?CareerCode={0}&StudentCode={1}&CourseCo
de={2}",


                                                                CareerCode,


                                                                StudentCode,


                                                                Code),
```

```
                                        UriKind.Relative)

                        )

                                    ));
            }
        }
    }
}
```

CourseStudentSituationViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseStudentSituationViewModel : ViewModel
    {
        /// <summary>
        /// The <see cref="PartialExamResult" /> property's name.
        /// </summary>
        public const string PartialExamResultPropertyName = "PartialExamResult";

        private double _partialExamResult = 0;

        /// <summary>
        /// Sets and gets the PartialExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double PartialExamResult
        {
            get
            {
                return _partialExamResult;
            }
            set
            {
                Set(PartialExamResultPropertyName, ref _partialExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="RecuperationExamResult" /> property's name.
        /// </summary>
        public const string RecuperationExamResultPropertyName =
"RecuperationExamResult";

        private double _recuperationExamResult = 0;

        /// <summary>
        /// Sets and gets the RecuperationExamResult property.
```

```
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double RecuperationExamResult
        {
            get
            {
                return _recuperationExamResult;
            }
            set
            {
                Set(() => RecuperationExamResult, ref _recuperationExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="FinalExamResult" /> property's name.
        /// </summary>
        public const string FinalExamResultPropertyName = "FinalExamResult";

        private double _finalExamResult = 0;

        /// <summary>
        /// Sets and gets the FinalExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double FinalExamResult
        {
            get
            {
                return _finalExamResult;
            }
            set
            {
                Set(() => FinalExamResult, ref _finalExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="PracticWorkExamResult" /> property's name.
        /// </summary>
        public const string PracticWorkExamResultPropertyName =
"PracticWorkExamResult";

        private double _practicWorkExamResult = 0;

        /// <summary>
        /// Sets and gets the PracticWorkExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double PracticWorkExamResult
        {
            get
            {
                return _practicWorkExamResult;
```

```
            }
            set
            {
                Set(() => PracticWorkExamResult, ref _practicWorkExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="TotalAssistedHours" /> property's name.
        /// </summary>
        public const string TotalAssistedHoursPropertyName = "TotalAssistedHours";

        private double _totalAssistedHours = 0;

        /// <summary>
        /// Sets and gets the TotalAssistedHours property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double TotalAssistedHours
        {
            get
            {
                return _totalAssistedHours;
            }
            set
            {
                Set(() => TotalAssistedHours, ref _totalAssistedHours, value);
            }
        }

        /// <summary>
        /// The <see cref="ActualAssistedPercent" /> property's name.
        /// </summary>
        public const string ActualAssistedPercentPropertyName = "ActualAssistedPercent";

        private double _actualAssistedPercent = 0.0;

        /// <summary>
        /// Sets and gets the ActualAssistedPercent property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double ActualAssistedPercent
        {
            get
            {
                return _actualAssistedPercent;
            }
            set
            {
                Set(() => ActualAssistedPercent, ref _actualAssistedPercent, value);
            }
        }
    }
}
```

CoursesViewModel.cs

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CoursesViewModel : ViewModel
    {
        public CoursesViewModel(IStudentRepository studentRepository)
        {
            Courses = new ObservableCollection<CourseResumeViewModel>();
            var student = studentRepository.GetStudent();

            if (student == null) return;

            var courses = student.Courses;
            if (courses != null)
                courses.ToList().ForEach(c => Courses.Add(new CourseResumeViewModel
                    {
                        Title = c.Course.Title,
                        Code = c.Course.Code,
                        Professor = c.Course.Professor,
                        Attendance = c.Attendance(),
                        StudentCode = student.StudentCode,
                        CareerCode = student.CareerCode
                    }));
        }


        public ObservableCollection<CourseResumeViewModel> Courses { get; private set; }


    }
}
```

CourseViewModel.cs

```
using System.Linq;
using System.Windows;
using System.Windows.Threading;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;
using System;
using TesinaMobileCloud.Phone.Services.Support;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
```

```
    public class CourseViewModel : ViewModel
    {
        private readonly IStudentCourseRepository _repository;
        private readonly ISyncronizationService _syncService;
        private readonly IStudentInformationManager _studentInformationManager;

        public CourseViewModel(IStudentCourseRepository repository,
ISyncronizationService syncService, IStudentInformationManager
studentInformationManager)
        {
            _repository = repository;
            _syncService = syncService;
            _studentInformationManager = studentInformationManager;
        }

        /// <summary>
        /// The <see cref="Title" /> property's name.
        /// </summary>
        public const string TitlePropertyName = "Title";
        private string _title = string.Empty;

        /// <summary>
        /// Sets and gets the Title property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Title
        {
            get
            {
                return _title;
            }
            set
            {
                Set(() => Title, ref _title, value);
            }
        }

        #region Description PivotItem
        /// <summary>
        /// The <see cref="CourseDescription" /> property's name.
        /// </summary>
        public const string CourseDescriptionPropertyName = "CourseDescription";

        private CourseDescriptionViewModel _myProperty = null;

        /// <summary>
        /// Sets and gets the CourseDescription property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public CourseDescriptionViewModel CourseDescriptionViewModel
        {
            get
            {
                return _myProperty;
```

```
    }

    set
    {
        if (_myProperty == value)
        {
            return;
        }

        RaisePropertyChanging(() => CourseDescriptionViewModel);
        _myProperty = value;
        RaisePropertyChanged(() => CourseDescriptionViewModel);
    }
}
#endregion

#region My Situation PivotItem
/// <summary>
/// The <see cref="CourseSituation" /> property's name.
/// </summary>
public const string CourseSituationPropertyName = "CourseSituation";

private CourseStudentSituationViewModel _courseStudent = null;

/// <summary>
/// Sets and gets the CourseSituation property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public CourseStudentSituationViewModel CourseStudentSituationViewModel
{
    get
    {
        return _courseStudent;
    }

    set
    {
        if (_courseStudent == value)
        {
            return;
        }

        RaisePropertyChanging(() => CourseStudentSituationViewModel);
        _courseStudent = value;
        RaisePropertyChanged(() => CourseStudentSituationViewModel);
    }
}


#endregion

public int Code { get; set; }
public int StudentCode { get; private set; }
public int CareerCode { get; set; }
```

```
        public override void OnNavigateTo(System.Collections.Generic.IDictionary<string,
string> queryString)
        {
            string code;
            string studentCode;
            string careerCode;

            if (queryString.TryGetValue("StudentCode", out studentCode))
                StudentCode = int.Parse(studentCode);
            if (queryString.TryGetValue("CareerCode", out careerCode))
                CareerCode = int.Parse(careerCode);

            if (queryString.TryGetValue("CourseCode", out code))
            {
                var course = _repository.GetCourse(code);
                Code = int.Parse(code);
                CourseDescriptionViewModel = new CourseDescriptionViewModel
                {
                    Description = course.Course.Description,
                    FinalExamDate = course.Course.FinalExamDate != null ?
((DateTime)course.Course.FinalExamDate).ToShortDateString() : string.Empty,
                    PartialExamDate = course.Course.PartialExamDate != null ?
((DateTime)course.Course.PartialExamDate).ToShortDateString() : string.Empty,
                    PracticWorkExamDate = course.Course.PracticWorkPresentationDate != null ?
((DateTime)course.Course.PracticWorkPresentationDate).ToShortDateString() :
string.Empty,
                    Professor = course.Course.Professor,
                    RecuperationExamDate = course.Course.RecuperationExameDate != null ?
((DateTime)course.Course.RecuperationExameDate).ToShortDateString() : string.Empty,
                    Scheduled = course.Course.Scheduled,
                    TotalHoursOfClasses = course.Course.TotalHoursOfClasses
                };
                CourseStudentSituationViewModel = new CourseStudentSituationViewModel
                {
                    ActualAssistedPercent = course.Attendance(),
                    FinalExamResult = course.FinalExamResult,
                    PartialExamResult = course.PartialExamResult,
                    RecuperationExamResult = course.RecuperationExamResult,
                    PracticWorkExamResult = course.PracticWorkResult,
                    TotalAssistedHours = course.TotalHoursAssisted

                };
                Title = course.Course.Title;

                SyncCourseInfo.Execute(null);
            }
        }

        public override void OnNavigateFrom(System.Collections.Generic.IDictionary<string,
string> queryString)
        {
            var course = _repository.GetCourse(Code.ToString());
```

```
            var command = new RelayCommand(() =>
_studentInformationManager.AddCourseExamsDatesReminders(course).Subscribe(
                        x => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; }),
                        ex => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false;
}),
                        () => Deployment.Current.Dispatcher.BeginInvoke(() =>
                        {
                            Updating = false;
                            base.OnNavigateFrom(queryString);
                        })));

            command.Execute(null);
        }

        private RelayCommand _syncCourseInfo;

        /// <summary>
        /// Gets the MyCommand.
        /// </summary>
        public RelayCommand SyncCourseInfo
        {
            get
            {
                Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; });
                return _syncCourseInfo
                    ?? (_syncCourseInfo = new RelayCommand(() =>
_syncService.SyncCourse(Code, StudentCode.ToString(), CareerCode.ToString()
).Subscribe(
                        x => Deployment.Current.Dispatcher.BeginInvoke(() =>
                            {
                                //DescriptionViewModel
                                CourseDescriptionViewModel.Description = x.Course.Description;
                                CourseDescriptionViewModel.FinalExamDate =
x.Course.FinalExamDate != null ?
((DateTime)x.Course.FinalExamDate).ToShortDateString() : string.Empty;
                                CourseDescriptionViewModel.PartialExamDate =
x.Course.PartialExamDate != null ?
((DateTime)x.Course.PartialExamDate).ToShortDateString() : string.Empty;
                                CourseDescriptionViewModel.Professor = x.Course.Professor;
                                CourseDescriptionViewModel.PracticWorkExamDate =
x.Course.PracticWorkPresentationDate != null ?
((DateTime)x.Course.PracticWorkPresentationDate).ToShortDateString() : string.Empty;
                                CourseDescriptionViewModel.Scheduled = x.Course.Scheduled;
                                CourseDescriptionViewModel.RecuperationExamDate =
x.Course.RecuperationExameDate != null ?
((DateTime)x.Course.RecuperationExameDate).ToShortDateString() : string.Empty;

                                //MySituationViewModel
                                CourseStudentSituationViewModel.ActualAssistedPercent =
x.Attendance();
                                CourseStudentSituationViewModel.FinalExamResult =
x.FinalExamResult;
                                CourseStudentSituationViewModel.PartialExamResult =
x.PartialExamResult;
```

```
                        CourseStudentSituationViewModel.PracticWorkExamResult =
x.PracticWorkResult;
                        CourseStudentSituationViewModel.RecuperationExamResult =
                            x.RecuperationExamResult;
                        CourseStudentSituationViewModel.TotalAssistedHours =
x.TotalHoursAssisted;
                    }),
                (Exception ex) => Deployment.Current.Dispatcher.BeginInvoke(() => {
Updating = false; }),
                () => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false;
}));
            }
        }


    }
}
```

MainViewModel.cs

```
using System;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class MainViewModel : ViewModel
    {
        private readonly IStudentRepository _studentRepository;

        public MainViewModel(IStudentRepository studentRepository)
        {
            _studentRepository = studentRepository;

            var student = _studentRepository.GetStudent();
            if (student != null)
            {
                StudentCode = student.StudentCode;
                CareerCode = student.CareerCode.ToString();
            }

        }

        public string CareerCode { get; set; }
        /// <summary>
        /// The <see cref="StudentCode" /> property's name.
        /// </summary>
        public const string StudentCodePropertyName = "StudentCode";

        private int _studentCode = 0;
        /// <summary>
        /// Sets and gets the StudentCode property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
```

```
        public int StudentCode
        {
            get
            {
                return _studentCode;
            }

            set
            {
                if (_studentCode == value)
                {
                    return;
                }

                RaisePropertyChanging(() => StudentCode);
                _studentCode = value;
                RaisePropertyChanged(() => StudentCode);
            }
        }

        private RelayCommand _configurationPageNavigation;

        /// <summary>
        /// Gets the MyCommand.
        /// </summary>
        public RelayCommand NavigateToSettingsView
        {
            get
            {
                return _configurationPageNavigation
                    ?? (_configurationPageNavigation = new RelayCommand(() =>
NavigationService.NavigateTo(new
Uri(String.Format("/View/SettingsView.xaml?StudentCode={0}&CareerCode={1}",
StudentCode, CareerCode),
                                                                        UriKind.Relative)
                                                    )));
            }
        }

        public override void OnNavigateTo(System.Collections.Generic.IDictionary<string,
string> queryString)
        {
            string careerCode;
            string studentCode;
            string courseCode;
            queryString.TryGetValue("StudentCode", out studentCode);
            queryString.TryGetValue("CareerCode", out careerCode);
            queryString.TryGetValue("CourseCode", out courseCode);

            if (string.IsNullOrEmpty(careerCode) || string.IsNullOrEmpty(careerCode) ||
string.IsNullOrEmpty(careerCode))
            {
                base.OnNavigateTo(queryString);
            }
```

```
                else
                {

                    NavigationService.NavigateTo(
                        new Uri(String.Format(

            "/Views/CourseView.xaml?CareerCode={0}&amp;amp;StudentCode={1}&amp;amp;CourseCode={2}"
            ,
                            careerCode,
                            studentCode,
                            courseCode)));
                }
            }
        }
    }
```

SettingsViewModel.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Windows;
    using System.Windows.Threading;
    using GalaSoft.MvvmLight.Command;
    using Microsoft.Phone.Notification;
    using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
    using TesinaMobileCloud.Phone.Services.Interfaces;
    using TesinaMobileCloud.Phone.Services.Support;

    namespace TesinaMobileCloud.Phone.Client.ViewModel
    {
        public class SettingsViewModel : ViewModel
        {
            private readonly IStudentRepository _studentRepository;
            private readonly ISyncronizationService _syncronizationService;

            public SettingsViewModel(IStudentRepository studentRepository,
    ISyncronizationService syncronizationService)
            {
                _studentRepository = studentRepository;
                _syncronizationService = syncronizationService;
                var student = _studentRepository.GetStudent();

                if (student != null)
                    if (!string.IsNullOrEmpty(student.DeviceUri))
                        _isChecked = 1;
            }

            /// <summary>
            /// The <see cref="StudentCode" /> property's name.
            /// </summary>
            public const string StudentCodePropertyName = "StudentCode";
            private string _studentCode = string.Empty;
            /// <summary>
```

```
/// Sets and gets the StudentCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string StudentCode
{
    get
    {
        return _studentCode;
    }

    set
    {
        if (_studentCode == value)
        {
            return;
        }

        RaisePropertyChanging(StudentCodePropertyName);
        _studentCode = value;
        RaisePropertyChanged(StudentCodePropertyName);
    }
}

/// <summary>
/// The <see cref="IsChecked" /> property's name.
/// </summary>
public const string IsCheckedPropertyName = "IsChecked";
private int? _isChecked = null;
/// <summary>
/// Sets and gets the IsChecked property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int? IsChecked
{
    get
    {
        return _isChecked;
    }

    set
    {
        if (_isChecked == value)
        {
            return;
        }

        RaisePropertyChanging(IsCheckedPropertyName);
        _isChecked = value;
        RaisePropertyChanged(IsCheckedPropertyName);
    }
}

/// <summary>
/// The <see cref="CareerCode" /> property's name.
```

```
/// </summary>
public const string CareerCodePropertyName = "CareerCode";
private string _careerCode = string.Empty;
/// <summary>
/// Sets and gets the CareerCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string CareerCode
{
    get
    {
        return _careerCode;
    }

    set
    {
        if (_careerCode == value)
        {
            return;
        }

        RaisePropertyChanging(CareerCodePropertyName);
        _careerCode = value;
        RaisePropertyChanged(CareerCodePropertyName);
    }
}

private RelayCommand _saveChangesInSettings;
/// <summary>
/// Gets the MyCommand.
/// </summary>
public RelayCommand SaveChangesInSettings
{
    get
    {
        return _saveChangesInSettings
            ?? (_saveChangesInSettings = new RelayCommand(
                                () =>
                                {
                                    if (IsChecked != 1) return;
                                    GetNotificationUri();
                                }));
    }
}

private void GetNotificationUri()
{
    const string channelName = "SiaMobileChannel";

    var pushChannel = HttpNotificationChannel.Find(channelName);

    // If the channel was not found, then create a new connection to the push service.
    if (pushChannel == null)
    {
```

```
            pushChannel = new HttpNotificationChannel(channelName);

            // Register for all the events before attempting to open the channel.
            pushChannel.ChannelUriUpdated += PushChannel_ChannelUriUpdated;
            pushChannel.ErrorOccurred += PushChannel_ErrorOccurred;

            pushChannel.Open();
            // Bind this new channel for Tile events.
            pushChannel.BindToShellTile();
            pushChannel.BindToShellToast();
        }
        else
        {
            // The channel was already open, so just register for all the events.
            pushChannel.ChannelUriUpdated += PushChannel_ChannelUriUpdated;
            pushChannel.ErrorOccurred += PushChannel_ErrorOccurred;

            if (!pushChannel.IsShellTileBound) pushChannel.BindToShellTile();
            if (!pushChannel.IsShellToastBound) pushChannel.BindToShellToast();
        }
    }

    void PushChannel_ChannelUriUpdated(object sender,
NotificationChannelUriEventArgs e)
    {
        Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; });
        _syncronizationService.RegisterNotificationChannelForStudent(StudentCode,
CareerCode, e.ChannelUri.ToString()).Subscribe(
                                    (TaskSummary x) =>
                                    {
                                        if (x.Result == TaskResult.Failed)
                                        {
                                            Deployment.Current.Dispatcher.BeginInvoke(() =>
MessageBox.Show("Se ha producido un error al tratar de guardar la configuración"));
                                        }
                                    },
                                    (Exception ex) =>
Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false; }),
                                    () => Deployment.Current.Dispatcher.BeginInvoke(() => {
Updating = false; }));
    }

    void PushChannel_ErrorOccurred(object sender, NotificationChannelErrorEventArgs
e)
    {
        Deployment.Current.Dispatcher.BeginInvoke(() => MessageBox.Show("Se ha
producido un error al tratar de guardar la configuración"));
    }



    public override void OnNavigateTo(IDictionary<string, string> queryString)
    {
        string studentCode;
```

```
            string courseCode;

            if (queryString.TryGetValue("StudentCode", out studentCode))
            {
                _studentCode = studentCode;
            }

            if (queryString.TryGetValue("CareerCode", out courseCode))
            {
                _careerCode = courseCode;
            }

            base.OnNavigateTo(queryString);
        }
    }
}
```

ViewModel.cs

```
using System.Collections.Generic;
using GalaSoft.MvvmLight;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;
using GalaSoft.MvvmLight.Ioc;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public abstract class ViewModel : ViewModelBase
    {
        protected readonly INavigationService NavigationService;

        protected ViewModel()
        {
            NavigationService = SimpleIoc.Default.GetInstance<INavigationService>();
        }

        /// <summary>
        /// The <see cref="Updating" /> property's name.
        /// </summary>
        public const string UpdatingPropertyName = "Updating";

        private bool _updating = false;

        /// <summary>
        /// Sets and gets the Updating property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public bool Updating
        {
            get
            {
                return _updating;
            }

            set
```

```
            {
                if (_updating == value)
                {
                    return;
                }

                RaisePropertyChanging(UpdatingPropertyName);
                _updating = value;
                RaisePropertyChanged(UpdatingPropertyName);
            }
        }

        public virtual void OnNavigateTo(IDictionary<string, string> queryString)
        {
        }

        public virtual void OnNavigateFrom(IDictionary<string, string> queryString)
        {
        }
    }
}
```

ViewModelLocator.cs

```
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;
using TesinaMobileCloud.Phone.Client.ViewServices;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;
using TesinaMobileCloud.Phone.Data.Repositories;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class ViewModelLocator
    {
        static ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            // Create run time view services and models
            SimpleIoc.Default.Register<IStudentCourseRepository,
StudentCourseRepository>();
            SimpleIoc.Default.Register<IStudentRepository, StudentRepository>();
            SimpleIoc.Default.Register<IScheduleActionClient, ScheduleActionClient>();
            SimpleIoc.Default.Register<IScheduleActionService,
ScheduleActionServiceAdapter>();
            SimpleIoc.Default.Register<INavigationService, NavigationService>();
            SimpleIoc.Default.Register<ISyncronizationService, SyncronizationService>();
            SimpleIoc.Default.Register<ICloudService, CloudService>();
            SimpleIoc.Default.Register<IStudentInformationManager,
StudentInformationManager>();
```

```
            SimpleIoc.Default.Register<MainViewModel>();
            SimpleIoc.Default.Register<SettingsViewModel>();
            SimpleIoc.Default.Register<CoursesViewModel>();
            SimpleIoc.Default.Register<CourseResumeViewModel>();
            SimpleIoc.Default.Register<CourseViewModel>();
            SimpleIoc.Default.Register<CourseStudentSituationViewModel>();
            SimpleIoc.Default.Register<CourseDescriptionViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }
        public CoursesViewModel Courses
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CoursesViewModel>();
            }
        }
        public CourseResumeViewModel CourseResume
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CourseResumeViewModel>();
            }

        }
        public CourseViewModel Course
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CourseViewModel>();
            }

        }
        public CourseDescriptionViewModel CourseDescription
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CourseDescriptionViewModel>();
            }

        }
        public CourseStudentSituationViewModel CourseStudentSituation
        {
            get
            {
                return
ServiceLocator.Current.GetInstance<CourseStudentSituationViewModel>();
```

```
            }

        }
        public IScheduleActionClient ScheduleActions
        {
            get
            {
                return ServiceLocator.Current.GetInstance<IScheduleActionClient>();
            }
        }

        public SettingsViewModel Settings
        {
            get { return ServiceLocator.Current.GetInstance<SettingsViewModel>(); }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

ViewServices

NavigationService.cs

```
using Microsoft.Phone.Controls;
using System;
using System.Windows;
using System.Windows.Navigation;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewServices
{
    public class NavigationService : INavigationService
    {
        private PhoneApplicationFrame _mainFrame;

        public event NavigatingCancelEventHandler Navigating;

        public void NavigateTo(Uri pageUri)
        {
            if (InitializatedFrame())
            {
                _mainFrame.Navigate(pageUri);
            }
        }

        public void GoBack()
        {
            if (InitializatedFrame() && _mainFrame.CanGoBack)
            {
                _mainFrame.GoBack();
            }
```

```
        }

        private bool InitializatedFrame()
        {
            if (_mainFrame != null)
                return true;

            _mainFrame = Application.Current.RootVisual as PhoneApplicationFrame;

            if (_mainFrame != null)
            {
                _mainFrame.Navigating += (s, e) =>
                {
                    if (Navigating != null)
                    {
                        Navigating(s, e);
                    }
                };

                return true;
            }

            return false;
        }
    }
}
```

Interfaces

INavigationService.cs

```
using System;
using System.Windows.Navigation;

namespace TesinaMobileCloud.Phone.Client.ViewServices.Interfaces
{
    public interface INavigationService
    {
        event NavigatingCancelEventHandler Navigating;
        void NavigateTo(Uri pageUri);
        void GoBack();
    }

}
```

TesinaMobileCloud.Phone.Client.Test

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.Test.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test.ViewModel" mc:Ignorable="d">
  <!--Application Resources-->
  <Application.Resources>
    <ResourceDictionary>
      <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
      <ResourceDictionary.MergedDictionaries></ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
  <Application.ApplicationLifetimeObjects>
    <!--Required object that handles lifetime events for the application-->
    <shell:PhoneApplicationService Launching="Application_Launching"
Closing="Application_Closing" Activated="Application_Activated"
Deactivated="Application_Deactivated" />
  </Application.ApplicationLifetimeObjects>
</Application>
```

App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
```

```
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
                // application's PhoneApplicationService object to Disabled.
                // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
                // and consume battery power when the user is not using the phone.
                PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
            }

        }

        // Code to execute when the application is launching (eg, from Start)
        // This code will not execute when the application is reactivated
        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
        }

        // Code to execute when the application is activated (brought to foreground)
        // This code will not execute when the application is first launched
        private void Application_Activated(object sender, ActivatedEventArgs e)
        {
        }

        // Code to execute when the application is deactivated (sent to background)
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEventArgs e)
        {
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
        {
        }
```

```
        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // A navigation has failed; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;
```

```
            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="TesinaMobileCloud.Phone.Client.Test.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
```

```
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Testing;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            this.Content = UnitTestSystem.CreateTestPage();
        }
    }
}
```

UnitTest

CoursesViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.ObjectModel;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CoursesViewModelFixture
    {
        readonly IStudentRepository _studentRepository = new MockStudentRepository();

        [TestMethod]
        public void CoursesViewModel()
        {
            var sut = new CoursesViewModel(_studentRepository);

            Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
            Assert.IsInstanceOfType(sut.Courses,
typeof(ObservableCollection<CourseResumeViewModel>));
        }

        [TestMethod]
        public void CoursesViewModelFillCoursesList()
        {
            var sut = new CoursesViewModel(_studentRepository);

            Assert.IsNotNull(sut.Courses);
            Assert.AreEqual(1, sut.Courses.Count);
```

```
        }
    }
}
```

CourseViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Windows.Media;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CourseViewModelFixture
    {

        [TestMethod]
        public void CourseViewModelWithParams()
        {
            var sut = new CourseResumeViewModel
            {
                Code = 124,
                Title = "Title",
                Professor = "Prof",
                //YearOfCareer = 1,
                Attendance = 1
            };

            Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
            Assert.IsNotNull(sut);
            Assert.AreEqual(124, sut.Code);
            Assert.AreEqual("Title", sut.Title);
            //Assert.AreEqual(1, sut.YearOfCareer);
            Assert.AreEqual("Prof", sut.Professor);
            Assert.AreEqual(1, sut.Attendance);
            Assert.IsInstanceOfType(sut.AttendanceState, typeof(SolidColorBrush));
        }

        [TestMethod]
        public void CourseViewModelWithOutParams()
        {
            var sut = new CourseResumeViewModel();
            Assert.IsNotNull(sut);
        }
    }
}
```

ViewModel

MainViewModel.cs

```
using GalaSoft.MvvmLight;
```

```
namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
    /// <summary>
    /// This class contains properties that the main View can data bind to.
    /// <para>
    /// Use the <strong>mvvminpc</strong> snippet to add bindable properties to this
ViewModel.
    /// </para>
    /// <para>
    /// You can also use Blend to data bind with the tool's support.
    /// </para>
    /// <para>
    /// See http://www.galasoft.ch/mvvm
    /// </para>
    /// </summary>
    public class MainViewModel : ViewModelBase
    {
        /// <summary>
        /// Initializes a new instance of the MainViewModel class.
        /// </summary>
        public MainViewModel()
        {
            ////if (IsInDesignMode)
            ////{
            ////    // Code runs in Blend --> create design time data.
            ////}
            ////else
            ////{
            ////    // Code runs "for real"
            ////}
        }
    }
}
```

ViewModelLocator.cs

```
/*
  In App.xaml:
  <Application.Resources>
      <vm:ViewModelLocator xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test"
                           x:Key="Locator" />
  </Application.Resources>

  In the View:
  DataContext="{Binding Source={StaticResource Locator}, Path=ViewModelName}"

  You can also use Blend to do all this with the tool's support.
  See http://www.galasoft.ch/mvvm
*/

using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;
```

```
namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
    /// <summary>
    /// This class contains static references to all the view models in the
    /// application and provides an entry point for the bindings.
    /// </summary>
    public class ViewModelLocator
    {
        /// <summary>
        /// Initializes a new instance of the ViewModelLocator class.
        /// </summary>
        public ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            ////if (ViewModelBase.IsInDesignModeStatic)
            ////{
            ////    // Create design time view services and models
            ////    SimpleIoc.Default.Register<IDataService, DesignDataService>();
            ////}
            ////else
            ////{
            ////    // Create run time view services and models
            ////    SimpleIoc.Default.Register<IDataService, DataService>();
            ////}

            SimpleIoc.Default.Register<MainViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

TesinaMobileCloud.Phone.Data

Repositories

MockStudentCourseRepository.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
```

```
namespace TesinaMobileCloud.Phone.Data.Repositories
{
   public class MockStudentCourseRepository : IStudentCourseRepository
   {
      private static List<StudentCourse> _courses;

      public MockStudentCourseRepository()
      {
         _courses = new List<StudentCourse>
         {
            new StudentCourse
            {
               Course = new Course
               {
                  Code = 1,
                  Title = "Programación I",
                  Professor = "Ing. Carlos Rodrigo",
                  Description = "Programacion I introduce los conceptos basicos de la
programacion, "+
                  "comenzado con el paradigma estructurado y luego introduciendo el
paradigma de objetos. "+
                  "El lenguaje de programacion a utilizar es Java.",
                  TotalHoursOfClasses = 60,
                  Scheduled = "Miercoles 17Hs",
                  FinalExamDate =new DateTime(2013,02,4,00,22,00),
                  PartialExamDate = new DateTime(2013,02,4,00,21,00),
                  PracticWorkPresentationDate = DateTime.MinValue,
                  RecuperationExameDate = DateTime.MinValue
               },
               FinalExamResult = 10,
               PartialExamResult = 9,
               PracticWorkResult = 9,
               RecuperationExamResult = 0,
               TotalHoursAssisted = 40
            }
         };
      }

      public IEnumerable<StudentCourse> GetCourses()
      {
         return _courses;
      }

      public void AddCourses(IEnumerable<StudentCourse> courses)
      {
         _courses.Clear();
         foreach (var course in courses)
            _courses.Add(course);
      }

      public void AddCourse(StudentCourse course)
      {
         throw new NotImplementedException();
```

```
        }

        public StudentCourse GetCourse(string code)
        {
            return new StudentCourse
              {
                 Course = new Course
                    {
                       Code = 1,
                       Title = "Programación I",
                       Professor = "Ing. Carlos Rodrigo",
                       Description = "Programacion I introduce los conceptos basicos de la
programacion, " +
                                     "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. " +
                                     "El lenguaje de programacion a utilizar es Java.",
                       TotalHoursOfClasses = 60,
                       Scheduled = "Miercoles 17Hs",
                       FinalExamDate = new DateTime(2013, 03, 6, 00, 28, 00),
                       PartialExamDate = new DateTime(2013, 03, 6, 00, 26, 00),
                       PracticWorkPresentationDate = DateTime.MinValue,
                       RecuperationExameDate = DateTime.MinValue
                    },
                 FinalExamResult = 10,
                 PartialExamResult = 9,
                 PracticWorkResult = 9,
                 RecuperationExamResult = 0,
                 TotalHoursAssisted = 40
              };
        }
    }

}
```

StudentCourseRepository.cs

```
using System.Collections.Generic;
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class StudentCourseRepository : IStudentCourseRepository
    {
        private const string studentcourseFormat = "StudentCourse{0}";
        private readonly IsolatedStorageSettings _isolatedStorage;

        public StudentCourseRepository()
        {
            _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
        }

        public IEnumerable<StudentCourse> GetCourses()
```

```
        {
            IEnumerable<StudentCourse> courses;
            _isolatedStorage.TryGetValue("StudentCourse", out courses);
            return courses;
        }

        public void AddCourses(IEnumerable<StudentCourse> courses)
        {
            foreach (var course in courses)
            {
                AddCourseToIsolatedStorage(course);
            }
            _isolatedStorage.Save();
        }

        public void AddCourse(StudentCourse course)
        {
            AddCourseToIsolatedStorage(course);
            _isolatedStorage.Save();
        }

        public StudentCourse GetCourse(string code)
        {
            StudentCourse course;
            var courseCode = string.Format(studentcourseFormat, code);
            _isolatedStorage.TryGetValue(courseCode, out course);
            return course;
        }

        private void AddCourseToIsolatedStorage(StudentCourse course)
        {
            StudentCourse studentCourse;
            var courseCode = string.Format(studentcourseFormat, course.Course.Code);
            if (_isolatedStorage.TryGetValue(courseCode, out studentCourse))
            {
                _isolatedStorage.Remove(courseCode);
            }
            _isolatedStorage.Add(courseCode, course);
        }
    }
}
```

StudentRepository.cs

```
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class StudentRepository : IStudentRepository
    {
        private readonly IsolatedStorageSettings _isolatedStorage;

        public StudentRepository()
```

```
        {
            _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
        }

        public void AddStudent(Student student)
        {
            Student studentToSave;
            if (_isolatedStorage.TryGetValue("Student", out studentToSave))
            {
                _isolatedStorage.Remove("Student");
            }
            _isolatedStorage.Add("Student", student);
            _isolatedStorage.Save();
        }

        public Student GetStudent()
        {
            Student student;
            _isolatedStorage.TryGetValue("Student", out student);
            return student;
        }
    }
}
```

Interfaces

IStudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Repositories.Interfaces
{
    public interface IStudentCourseRepository
    {
        void AddCourses(IEnumerable<StudentCourse> courses);
        void AddCourse(StudentCourse course);
        StudentCourse GetCourse(string code);
    }
}
```

IStudentRepository.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Repositories.Interfaces
{
    public interface IStudentRepository
    {
        void AddStudent(Student student);
        Student GetStudent();
    }
```

```
    public class MockStudentRepository : IStudentRepository
    {
        public void AddStudent(Student student)
        {
            throw new System.NotImplementedException();
        }

        public Student GetStudent()
        {
            return new Student
                {
                    Courses = new List<StudentCourse>
                                {
                                    new StudentCourse
                                    {
                                        Course = new Course
                                        {
                                            Code = 1,
                                            Title = "Programación I",
                                            Professor = "Ing. Carlos Rodrigo",
                                            Description = "Programacion I introduce los conceptos basicos
de la programacion, "+
                                            "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. "+
                                            "El lenguaje de programacion a utilizar es Java.",
                                            TotalHoursOfClasses = 60,
                                            Scheduled = "Miercoles 17Hs",
                                            FinalExamDate = new DateTime(2013,02,4,00,20,00),
                                            PartialExamDate = new DateTime(2013,02,4,00,21,00),
                                            PracticWorkPresentationDate = DateTime.MinValue,
                                            RecuperationExameDate = DateTime.MinValue
                                        },
                                        FinalExamResult = 10,
                                        PartialExamResult = 9,
                                        PracticWorkResult = 9,
                                        RecuperationExamResult = 0,
                                        TotalHoursAssisted = 40
                                    }
                                },
                    CareerCode = 10004,
                    StudentCode = 502,
                    FirstName = "Carlos Sergio",
                    LastName = "Rodrigo"
                };
        }
    }
}
```

TesinaMobileCloud.Phone.Data.Test

UnitTest1.cs

```
using System.Collections.Generic;
using System.Linq;
```

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Test
{
   [TestClass]
   public class StudentRepositoryFixture
   {
      [TestMethod]
      public void GetCourses()
      {
         var sut = new Data.Repositories.MockStudentCourseRepository();

         var result = sut.GetCourses();

         Assert.IsNotNull(result);
         Assert.IsInstanceOfType(result, typeof(IEnumerable<StudentCourse>));
      }

      [TestMethod]
      public void AddCourses()
      {
         var courses = new List<StudentCourse>
                     {
                        new StudentCourse
                        {
                           Course = new Course{
                           Title = "Programacion I", Professor = "Prof. Diana Ciccinelli"}
                        },
                        new StudentCourse
                        {
                           Course = new Course{
                           Title = "Programacion II", Professor = "Prof. Martin Cernadas"}
                        }
                     };
         var sut = new Data.Repositories.MockStudentCourseRepository();

         sut.AddCourses(courses);

         var result = sut.GetCourses();
         Assert.IsNotNull(result);
         Assert.AreEqual(2, result.Count());
      }
   }
}
```

TesinaMobileCloud.Phone.Services

CloudService.cs

```
using System;
using System.IO;
using System.Net;
using System.Runtime.Serialization;
```

```
using System.Runtime.Serialization.Json;
using System.Text;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services
{
    public class CloudService : ICloudService
    {
        public IObservable<Student> GetAllCourses(Uri baseUri)
        {
            var relativeUri = String.Format("RetriveStudentByCodeAndCareer/{0}/{1}", "10004",
"502");//TODO: Alto HardCoding, que lo tome del IsolatedStorage

            var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
            request.Method = "GET";
            request.Accept = "application/json";

            return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
                .Select(
                    response =>
                    {
                        using (var responseStream = response.GetResponseStream())
                        {
                            var serializer = new DataContractJsonSerializer(typeof(Student));
                            return serializer.ReadObject(responseStream) as Student;
                        }
                    }
                );
        }

        public IObservable<StudentCourse> GetCourse(Uri baseUri, int code, string
studentCode, string careerCode)
        {
            var relativeUri = String.Format("RetriveStudentCourse/{0}/{1}/{2}", careerCode,
studentCode, code.ToString());
            var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
            request.Method = "GET";
            request.Accept = "application/json";

            return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
                .Select(
                    response =>
                    {
                        using (var responseStream = response.GetResponseStream())
                        {
                            var serializer = new DataContractJsonSerializer(typeof(StudentCourse));
                            return serializer.ReadObject(responseStream) as StudentCourse;
                        }
                    }
```

```
            );
        }

        public IObservable<TaskSummary> RegisterNotificationChannelForStudent(Uri
baseUri, string studentCode, string careerCode, string uri)
        {

            var uriToSend = HttpUtility.UrlEncode(uri);
            var relativeUri =
String.Format("RegisterPushNotificationChannelForStudent?studentCode={0}&uri={1}",
studentCode, uriToSend);
            var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
            request.Method = "GET";
            request.Accept = "application/json";

            return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
                .Select(
                    response => new TaskSummary
                        {
                            Result = TaskResult.Success
                        })
                .Catch((Exception ex) => Observable.Return(new TaskSummary
                    {
                        Result = TaskResult.Failed
                    }));
        }
    }
}
```

ScheduleActionClient.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionClient : IScheduleActionClient
    {
        private const string PeriodicTaskName = "SyncCourses";
        private readonly IScheduleActionService _scheduleActionService;

        public ScheduleActionClient(IScheduleActionService scheduleActionService)
        {
            _scheduleActionService = scheduleActionService;
        }

        public void AddPeriodicTask()
        {
            var periodicTask = new PeriodicTask(PeriodicTaskName)
                               {
                                   Description = "Syncronization Courses information Task"
                               };
            if (_scheduleActionService.Find(PeriodicTaskName) != null)
```

```
                _scheduleActionService.Remove(PeriodicTaskName);

            _scheduleActionService.Add(periodicTask);

        #if DEBUG
            _scheduleActionService.LaunchForTest(PeriodicTaskName,
        TimeSpan.FromMinutes(1));
        #endif
        }

        public void RemovePeriodicTask()
        {
            if (_scheduleActionService.Find(PeriodicTaskName) != null)
                _scheduleActionService.Remove(PeriodicTaskName);
        }
    }
}
```

ScheduleActionServiceAdapter.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionServiceAdapter : IScheduleActionService
    {
        public ScheduledAction Find(string taskName)
        {
            return ScheduledActionService.Find(taskName);
        }

        public void Add(ScheduledAction action)
        {
            ScheduledActionService.Add(action);
        }

        public void Remove(string taskName)
        {
            ScheduledActionService.Remove(taskName);
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            ScheduledActionService.LaunchForTest(actionName, delay);
        }
    }
}
```

StudentInformationManager.cs

```
using System;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Data.DTO;
```

```
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services
{
   public class StudentInformationManager : IStudentInformationManager
   {
      private readonly IStudentRepository _studentRepository;
      private readonly IStudentCourseRepository _studentCourseRepository;
      private readonly IScheduleActionService _scheduleActionService;

      public StudentInformationManager(IStudentRepository studentRepository,
IStudentCourseRepository studentCourseRepository, IScheduleActionService
scheduleActionService)
      {
         _studentRepository = studentRepository;
         _studentCourseRepository = studentCourseRepository;
         _scheduleActionService = scheduleActionService;
      }

      public string ProcessStudentInformation(Student student)
      {
         _studentRepository.AddStudent(student);
         foreach (var studentCourse in student.Courses)
         {
            ProcessStudentCourseInformation(studentCourse);
         }

         return string.Empty;
      }

      public void ProcessStudentCourseInformation(StudentCourse studentCourse)
      {
         _studentCourseRepository.AddCourse(studentCourse);
      }

      public IObservable<TaskSummary>
AddCourseExamsDatesReminders(StudentCourse studentCourse)
      {
         try
         {
            var courseTitle = studentCourse.Course.Title;
            if (studentCourse.Course.PartialExamDate != null)
               AddRemainderForCourseExam(String.Format("{0}ExamParcial",
studentCourse.Course.Code),
                                         String.Format("Examen Parcial de {0}", courseTitle),
                                         String.Format("En d�a de ma�ana se dicta el Examen
Parcial de {0}", courseTitle),
                                         studentCourse.Course.PartialExamDate,
                                         24);
            if (studentCourse.Course.RecuperationExameDate != null)
               AddRemainderForCourseExam(String.Format("{0}ExamRecuperatorio",
studentCourse.Course.Code),
```

```
                                String.Format("Examen Recuperatorio de {0}", courseTitle),
                                String.Format("En d�a de ma�ana se dicta el Examen
Recuperatorio de {0}", courseTitle),
                                studentCourse.Course.RecuperationExameDate,
                                24);
            if (studentCourse.Course.PracticWorkPresentationDate != null)

AddRemainderForCourseExam(String.Format("{0}PresentacionTrabajosPracticos",
studentCourse.Course.Code),
                                String.Format("Presentaci�n de Trabajos Practicos de {0}",
courseTitle),
                                String.Format("En d�a de ma�ana se realiza la
Presentaci�n de Trabajos Practicos de {0}", courseTitle),
                                studentCourse.Course.PracticWorkPresentationDate,
                                24);

            if (studentCourse.Course.FinalExamDate != null)
            {
                if (!studentCourse.FinalExamInscripted)
                {
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal1",
studentCourse.Course.Code),
                                    String.Format("Examen Final de {0}", courseTitle),
                                    String.Format("Recuerde inscribirse al Examen Final de
{0}", courseTitle),
                                    studentCourse.Course.FinalExamDate,
                                    96);
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal2",
studentCourse.Course.Code),
                                    String.Format("Examen Final de {0}", courseTitle),
                                    String.Format("Recuerde inscribirse al Examen Final de
{0}", courseTitle),
                                    studentCourse.Course.FinalExamDate,
                                    84);
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal3",
studentCourse.Course.Code),
                                    String.Format("Examen Final de {0}", courseTitle),
                                    String.Format("Recuerde inscribirse al Examen Final de
{0}", courseTitle),
                                    studentCourse.Course.FinalExamDate,
                                    73);
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal",
studentCourse.Course.Code),
                                String.Format("Examen Final de {0}", courseTitle),
                                String.Format("En d�a de ma�ana se dicta el Examen Final
de  {0}", courseTitle),
                                studentCourse.Course.FinalExamDate,
                                24);
                }
                else
                {
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal1",
courseTitle));
```

```
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal2",
courseTitle));
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal3",
courseTitle));
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal",
courseTitle));
                }


            }

            return Observable.Return(new TaskSummary
                                {
                                    Result = TaskResult.Success
                                });
        }
        catch (Exception)
        {
            return Observable.Return(new TaskSummary
                                {
                                    Result = TaskResult.Success
                                });
        }
    }

    private void AddRemainderForCourseExam(string remainderName, string title, string
content, DateTime? examDate, int hoursEarlyToRemaind)
    {
        var firstPartialReminder = remainderName;
        if (examDate == null) return;
        var scheduledAction = _scheduleActionService.Find(firstPartialReminder);
        var beginTime = ((DateTime)examDate).AddHours(-hoursEarlyToRemaind);
        var reminder = new Reminder(firstPartialReminder)
            {
                BeginTime = beginTime,
                Title = title,
                Content = content
            };
        if (scheduledAction != null)
        {
            if (!scheduledAction.BeginTime.Equals(beginTime))
            {
                _scheduleActionService.Remove(firstPartialReminder);
                _scheduleActionService.Add(reminder);
            }
        }
        else
        {
            _scheduleActionService.Add(reminder);
        }
    }
  }
}
```

SyncronizationService.cs

```
using System;
using System.Linq;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services
{
    public class SyncronizationService : ISyncronizationService
    {
        private readonly ICloudService _cloudService;
        private readonly IStudentInformationManager _studentInformationManager;
#if DEBUG
        private readonly Uri _baseUri = new Uri("http://127.0.0.2:8081/Service1.svc/");
#else
        private readonly Uri _baseUri = new Uri("http://157.56.178.10:8081/Service1.svc/");
#endif
        public SyncronizationService(ICloudService cloudService,
IStudentInformationManager studentInformationManager)
        {
            _cloudService = cloudService;
            _studentInformationManager = studentInformationManager;
        }

        public IObservable<TaskSummary> SyncCourses()
        {
            var resuls = _cloudService.GetAllCourses(_baseUri)
                .Select(student =>
                    {
                        var message =
_studentInformationManager.ProcessStudentInformation(student);
                        return new TaskSummary
                                {
                                    Result = TaskResult.Success
                                };
                    })
                .Catch((Exception exception) =>
                        {
                            throw exception;
                        });

            return resuls;
        }

        public IObservable<StudentCourse> SyncCourse(int code, string studentCode, string
careerCode)
        {
            var result = _cloudService.GetCourse(_baseUri, code, studentCode, careerCode)
                                .Select(course =>
                                    {
```

```
            _studentInformationManager.ProcessStudentCourseInformation(course);
                                        return course;
                                    })
                                .Catch((Exception exception) =>
                                {
                                    throw exception;
                                });
                return result;
            }

            public IObservable<TaskSummary> RegisterNotificationChannelForStudent(string
        studentCode, string careerCode, string uri)
            {
                var resuls = _cloudService.RegisterNotificationChannelForStudent(_baseUri,
        studentCode, careerCode, uri).Select(summary => summary);
                return resuls;
            }
        }
    }
```

Interfaces

ICloudService.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ICloudService
    {
        IObservable<Student> GetAllCourses(Uri baseUri);
        IObservable<StudentCourse> GetCourse(Uri baseUri, int code, string studentCode,
string careerCode);
        IObservable<TaskSummary> RegisterNotificationChannelForStudent(Uri baseUri,
string studentCode, string careerCode, string uri);

    }
}
```

IScheduleActionClient.cs

```
namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionClient
    {
        void AddPeriodicTask();
        void RemovePeriodicTask();
    }
}
```

IScheduleActionService.cs

```
using System;
```

```
using Microsoft.Phone.Scheduler;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionService
    {
        ScheduledAction Find(string taskName);
        void Add(ScheduledAction action);
        void Remove(string taskName);
        void LaunchForTest(string actionName, TimeSpan delay);
    }
}
```

IStudentInformationManager.cs

```
using System;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IStudentInformationManager
    {
        string ProcessStudentInformation(Student student);
        void ProcessStudentCourseInformation(StudentCourse studentCourse);
        IObservable<TaskSummary> AddCourseExamsDatesReminders(StudentCourse
studentCourse);
    }
}
```

ISyncronizationService.cs

```
using System;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ISyncronizationService
    {
        IObservable<TaskSummary> SyncCourses();
        IObservable<StudentCourse> SyncCourse(int code, string studentCode, string
careerCode);
        IObservable<TaskSummary> RegisterNotificationChannelForStudent(string
studentCode, string careerCode, string uri);
    }
}
```

Mocks

MockCloudService.cs

```
using Microsoft.Phone.Reactive;
using System;
using System.Collections.Generic;
```

```
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
   public class MockCloudService : ICloudService
   {
      public IObservable<Student> GetAllCourses(Uri baseUri)
      {
         return Observable.ToObservable(new List<Student>{new Student
            {
               Courses = new List<StudentCourse>
                     {
                        new StudentCourse
                        {
                           Course = new Course
                           {
                              Code = 1,
                              Title = "Programaci�n I",
                              Professor = "Ing. Carlos Rodrigo",
                              Description = "Programacion I introduce los conceptos basicos
de la programacion, "+
                              "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. "+
                              "El lenguaje de programacion a utilizar es Java.",
                              TotalHoursOfClasses = 60,
                              Scheduled = "Miercoles 17Hs",
                              FinalExamDate = new DateTime(2013,02,04,00,05,00),
                              PartialExamDate = new DateTime(2013,02,04,00,08,00),
                              PracticWorkPresentationDate = new
DateTime(2013,02,04,00,07,00),
                              RecuperationExameDate = new DateTime(2013,02,04,00,06,00)
                           },
                           FinalExamResult = 10,
                           PartialExamResult = 9,
                           PracticWorkResult = 9,
                           RecuperationExamResult = 0,
                           TotalHoursAssisted = 40
                        }
                     }
         }});
      }

      public IObservable<StudentCourse> GetCourse(Uri baseUri, int code, string
studentCode, string careerCode)
      {
         throw new NotImplementedException();
      }

      public IObservable<TaskSummary> RegisterNotificationChannelForStudent(Uri
baseUri, string studentCode, string careerCode, string uri)
      {
         throw new NotImplementedException();
```

```
        }
    }
}
```

MockScheduleActionService.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockScheduleActionService : IScheduleActionService
    {
        private PeriodicTask periodic;
        public ScheduledAction Find(string taskName)
        {
            return periodic;
        }

        public void Add(ScheduledAction action)
        {
            periodic = new PeriodicTask("MockAdd");
        }

        public void Remove(string taskName)
        {
            periodic = null;
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            //throw new NotImplementedException();
        }
    }
}
```

MockSyncronizationService.cs

```
using System;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services.Support;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockSyncronizationService : ISyncronizationService
    {
        public IObservable<TaskSummary> SyncCourses()
        {
            return Observable.Return(new TaskSummary
                                     {
                                         Result = TaskResult.Success
                                     });
        }
```

```
        public IObservable<StudentCourse> SyncCourse(int code, string studentCode, string
    careerCode)
        {
            throw new NotImplementedException();
        }

        public IObservable<TaskSummary> RegisterNotificationChannelForStudent(string
    studentCode, string careerCode, string uri)
        {
            throw new NotImplementedException();
        }
    }
}
```

Support

TaskResult.cs

```
namespace TesinaMobileCloud.Phone.Services.Support
{
    public enum TaskResult
    {
        Success,
        Failed
    }
}
```

TaskSummary.cs

```
namespace TesinaMobileCloud.Phone.Services.Support
{
    public class TaskSummary
    {
        public TaskResult Result { get; set; }
    }
}
```

TesinaMobileCloud.Phone.Services.Test

CloudServiceFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class CloudServiceFixture
    {
        [TestMethod]
        public void GetAllCourses()
        {
            var sut = new MockCloudService();
```

```
            var result = sut.GetAllCourses(new Uri("http://127.0.0.1"));
            Assert.IsNotNull(result);
        }
    }
}
```

ScheduleActionsServiceFixture.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class ScheduleActionsServiceFixture
    {
        public Interfaces.IScheduleActionService mockScheduleService = new
MockScheduleActionService();

        [TestMethod]
        public void RemoveAgent()
        {
            var sut = new ScheduleActionClient(mockScheduleService);

            sut.RemovePeriodicTask();

            Assert.IsNull(mockScheduleService.Find("MockRemove"));

        }

        [TestMethod]
        public void AddAgent()
        {
            var sut = new ScheduleActionClient(mockScheduleService);

            sut.AddPeriodicTask();

            Assert.IsNotNull(mockScheduleService.Find("MockAdd"));
        }

    }
}
```

SyncronizationServiceFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Phone.Services.Support;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class SyncronizationServiceFixture
```

```
    {
        [TestMethod]
        public void SyncCourses()
        {
            var sut = new MockSyncronizationService();

            var result = sut.SyncCourses();

            Assert.IsNotNull(result);
        }

        //[TestMethod]
        //public void SyncCourse()
        //{
        //    var code = 1;
        //    var sut = new MockSyncronizationService();

        //    var result = sut.SyncCourse(code);

        //    Assert.IsNotNull(result);
        //}
    }
}
```

TesinaMobileCloud.Phone.Shared

Class1.cs

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using GalaSoft.MvvmLight.Ioc;

namespace TesinaMobileCloud.Phone.Shared
{
    public class CustomIocContainer : SimpleIoc
    {
        static CustomIocContainer()
        {

        }
    }
}
```

TesinaMobileCloud.PushNotificationSenderRole

PushNotificationSenderRole.cs

```
using System;
using System.Diagnostics;
using System.Net;
using System.Text;
using System.Threading;
using Microsoft.WindowsAzure.ServiceRuntime;
using Ninject;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Shared;

namespace TesinaMobileCloud.PushNotificationSenderRole
{
    public class PushNotificationSenderRole : RoleEntryPoint
    {
        private IKernel _container;
        private IQueue<QueueNotificationMessage> queue;
        //private IQueue<string> _errors;
        public override void Run()
        {
            queue = _container.Get<IQueue<QueueNotificationMessage>>();
            //_errors = _container.Get<IQueue<string>>();

            while (true)
            {
                Thread.Sleep(10000);
                if (!queue.HasMessage) continue;
                var notificationMessage = queue.GetMessage();

                try
                {
                    SendNotification(notificationMessage);
                }
                catch (Exception ex)
                {
                    queue.AddMessage(notificationMessage);
                }
            }
        }

        private static void SendNotification(QueueNotificationMessage notificationMessage)
        {
            var sendNotificationRequest =
WebRequest.Create(notificationMessage.Destinatary);
            var notificationMessageInBytes = new
UTF8Encoding().GetBytes(notificationMessage.Payload);

            sendNotificationRequest.Method = WebRequestMethods.Http.Post;
            sendNotificationRequest.ContentType = "text/xml";
            sendNotificationRequest.ContentLength = notificationMessageInBytes.Length;
            sendNotificationRequest.Headers["X-MessageID"] = Guid.NewGuid().ToString();
```

```
            sendNotificationRequest.Headers.Add("X-WindowsPhone-Target",
notificationMessage.Type);
            sendNotificationRequest.Headers.Add("X-NotificationClass",
notificationMessage.Priority);


            using (var requestStream = sendNotificationRequest.GetRequestStream())
            {
               requestStream.Write(notificationMessageInBytes, 0,
notificationMessageInBytes.Length);
            }

            // Send the notification and get the response.
            var response = sendNotificationRequest.GetResponse();
            var notificationStatus = response.Headers["X-NotificationStatus"];
            var notificationChannelStatus = response.Headers["X-SubscriptionStatus"];
            var deviceConnectionStatus = response.Headers["X-DeviceConnectionStatus"];
            //if(!notificationStatus.Equals("Connected") ||
!deviceConnectionStatus.Equals("Received"))
        }

        public override bool OnStart()
        {
            // Set the maximum number of concurrent connections
            ServicePointManager.DefaultConnectionLimit = 12;

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.
            _container = new StandardKernel(new ServiceModule());


            return base.OnStart();
        }
    }
}
```

TesinaMobileCloud.Shared

ServiceModule.cs

```
using Microsoft.WindowsAzure;
using Ninject.Modules;
using TesinaMobileCloud.Data;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Shared
{
    public class ServiceModule : NinjectModule
    {
```

```
        public override void Load()
        {
            Bind<CloudStorageAccount>().ToMethod(context =>
CloudStorageConfiguration.GetCloudAccount("DataConnectionString"));

            Bind<ITable<Course>>().To<AzureTable<Course>>();
            Bind<ITable<Career>>().To<AzureTable<Career>>();

Bind<ITable<CareerCourseRelationship>>().To<AzureTable<CareerCourseRelationship>>(
);
            Bind<ITable<Student>>().To<AzureTable<Student>>();

Bind<ITable<StudentCourseRelationship>>().To<AzureTable<StudentCourseRelationship>
>();


Bind<IQueue<QueueNotificationMessage>>().To<NotificationQueue<QueueNotificationMe
ssage>>();
            Bind<IQueue<string>>().To<PushNotificationErrorsQueue<string>>();

            Bind<ICareerRepository>().To<CareerRepository>();
            Bind<ICourseRepository>().To<CourseRepository>();

Bind<ICourseCareerRelationshipRepository>().To<CareerCourseRelationshipRepository>()
;
            Bind<IStudentRepository>().To<StudentRepository>();
            Bind<IStudentCourseRepository>().To<StudentCourseRepository>();
        }
    }
}
```

5. Anexo 5 – Iteración 5

ServiceRoleTest

IServiceRoleFixture.cs

```
using System;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using Course = TesinaMobileCloud.Data.DTO.Course;
using CourseAzure = TesinaMobileCloud.Data.Model.Course;
using Student = TesinaMobileCloud.Data.Model.Student;
using TesinaMobileCloud.Data.DTO;

namespace ServiceRoleTest
{
   [TestClass]
   public class ServiceRoleFixture
   {
      const string CareerCode = "502";
      const string StudentCode = "10004";
      const int CourseCode = 120;

      private Mock<ICourseRepository> _repositoryCourse;
      private Mock<ICareerRepository> _repositoryCareer;
      private Mock<ICourseCareerRelationshipRepository> _repositoryRelationship;
      private Mock<IStudentRepository> _student;
      private Mock<IStudentCourseRepository> _studentCourse;

      [TestInitialize]
      public void Setup()
      {
         _repositoryCourse = new Mock<ICourseRepository>();
         _repositoryCareer = new Mock<ICareerRepository>();
         _repositoryRelationship = new Mock<ICourseCareerRelationshipRepository>();
         _repositoryCourse.Setup(m => m.GetCoursesByCareer(CareerCode)).Returns(new
List<TesinaMobileCloud.Data.Model.Course>
                                                {
                                                   new
TesinaMobileCloud.Data.Model.Course(CourseCode,"Programación I")
                                                   {
                                                      Professor = "Mg. Angeleri, Paula",
                                                   }
                                                });
```

```
            _repositoryRelationship.Setup(m =>
m.GetCareerYearOfCourseByCareer(int.Parse(CareerCode),
                                                            CourseCode))
                            .Returns(1);
            _student = new Mock<IStudentRepository>();
            _studentCourse = new Mock<IStudentCourseRepository>();
        }


        [TestMethod]
        public void RetriveStudentByStudentCodeAndCareer()
        {
            _student.Setup(m =>
m.GetStudentByCareerCodeAndStudentCode(It.IsAny<string>(), It.IsAny<string>()))
                    .Returns(new Student
                    {
                        CareerCode = 502,
                        StudentCode = 10004,
                        FirstName = "Carlos",
                        LastName = "Rodrigo"
                    });
            _studentCourse.Setup(m => m.RetriveCoursesOfStudent(It.IsAny<string>()))
                        .Returns(new List<StudentCourseRelationship>
                        {
                            new StudentCourseRelationship
                            {
                                CourseCode = 1,
                            }
                        });
            _repositoryCourse.Setup(m => m.GetSingleCourse(It.IsAny<int>()))
                            .Returns(new CourseAzure
                            {
                                Code = 1,
                                Title = "Ing. del Software",
                                Professor = "Mg. Paula Angeleri",
                                TotalHoursOfClasses = 60,
                                Description = "Lalo",
                                Scheduled = "Miercoles 17Hs",
                                FinalExamDate = new DateTime(2013, 12, 22),
                                PartialExameDate = new DateTime(2013, 10, 10),
                                PracticWorkPresentationDate = new DateTime(2013, 12, 20),
                                RecuperationExameDate = new DateTime(2013, 11, 20)
                            });
            _repositoryCareer.Setup(m => m.GetCareer(It.IsAny<string>())).Returns(new
TesinaMobileCloud.Data.Model.Career());
```

```
            var sut = new SiaService.SiaService(_repositoryCourse.Object,
_repositoryRelationship.Object, _student.Object, _studentCourse.Object,
_repositoryCareer.Object);

            var result = sut.RetriveStudentByCodeAndCareer(StudentCode, CareerCode);

            Assert.IsNotNull(result);
            Assert.IsNotNull(result.Courses);
            Assert.AreEqual(1, result.Courses.Count());
            Assert.IsNotNull(result.Courses.First().PartialExamResult);
            Assert.IsNotNull(result.Courses.First().FinalExamResult);
            Assert.IsNotNull(result.Courses.First().RecuperationExamResult);
            Assert.IsNotNull(result.Courses.First().PracticWorkResult);
            Assert.IsNotNull(result.Courses.First().TotalHoursAssisted);
            Assert.IsNotNull(result.Courses.First().Course.TotalHoursOfClasses);
            Assert.IsNotNull(result.Courses.First().Course.Title);
            Assert.IsNotNull(result.Courses.First().Course.Professor);
            Assert.IsNotNull(result.Courses.First().Course.Code);
            Assert.IsNotNull(result.Courses.First().Course.Scheduled);
            Assert.IsNotNull(result.Courses.First().Course.Description);
            Assert.IsNotNull(result.Courses.First().Course.PartialExamDate);
            Assert.IsNotNull(result.Courses.First().Course.FinalExamDate);
            Assert.IsNotNull(result.Courses.First().Course.RecuperationExameDate);
            Assert.IsNotNull(result.Courses.First().Course.PracticWorkPresentationDate);
            Assert.IsNotNull(result.Career);
        }

        [TestMethod]
        public void RetriveStudentCourse()
        {
            var studentCode ="10004";
            var courseCode = "22";
            var careerCode = "502";
            _studentCourse.Setup(m => m.RetriveStudentCourseByCode(It.IsAny<string>(),
It.IsAny<string>()))
                          .Returns(
                               new StudentCourseRelationship
                                  {
                                     CourseCode = 1,
                                  }
                           );
            _repositoryCourse.Setup(m => m.GetSingleCourse(It.IsAny<int>()))
                           .Returns(new CourseAzure
                           {
                              Code = 1,
                              Title = "Ing. del Software",
                              Professor = "Mg. Paula Angeleri",
```

```
                    TotalHoursOfClasses = 60,
                    Description = "Lalo",
                    Scheduled = "Miercoles 17Hs",
                    FinalExamDate = new DateTime(2013, 12, 22),
                    PartialExameDate = new DateTime(2013, 10, 10),
                    PracticWorkPresentationDate = new DateTime(2013, 12, 20),
                    RecuperationExameDate = new DateTime(2013, 11, 20)
                });

            var sut = new SiaService.SiaService(_repositoryCourse.Object,
                                    _repositoryRelationship.Object,
                                    _student.Object, _studentCourse.Object,
                                    _repositoryCareer.Object);

            var result = sut.RetriveStudentCourse(careerCode, studentCode, courseCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(StudentCourse));
        }
    }
}
```

SiaService

AzureLocalStorageTraceListener.cs

```
using System;
using System.Diagnostics;
using System.IO;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class AzureLocalStorageTraceListener : XmlWriterTraceListener
    {
        public AzureLocalStorageTraceListener()
            : base(Path.Combine(AzureLocalStorageTraceListener.GetLogDirectory().Path,
"SiaService.svclog"))
        {
        }

        public static DirectoryConfiguration GetLogDirectory()
        {
            var directory = new DirectoryConfiguration
                                    {
                                        Container = "wad-tracefiles",
```

```
                                        DirectoryQuotaInMB = 10,
                                        Path =

RoleEnvironment.GetLocalResource("SiaService.svclog").RootPath
                                    };
            return directory;
        }
    }
}
```

Global.asax.cs

```
using System;
using Ninject;
using Ninject.Web.Common;
using TesinaMobileCloud.Shared;

namespace SiaService
{
    public class Global : NinjectHttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            Application_Start();
        }

        protected override IKernel CreateKernel()
        {
            return new StandardKernel(new ServiceModule());
        }
    }
}
```

ISiaService.cs

```
using System;
using System.ServiceModel;
using System.ServiceModel.Web;
using TesinaMobileCloud.Data.DTO;
using Student = TesinaMobileCloud.Data.DTO.Student;

namespace SiaService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "ISiaService" in both code and config file together.
    [ServiceContract]
    public interface ISiaService
    {
```

```
        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RetriveStudentByCodeAndCareer/{studentCode}/{careerCode}")]
        Student RetriveStudentByCodeAndCareer(string studentCode, string careerCode);

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RetriveStudentCourse/{careerCode}/{studentCode}/{courseCode}")]
        StudentCourse RetriveStudentCourse(string careerCode, string studentCode, string
courseCode);

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RegisterPushNotificationChannelForStudent?studentCode={studentCode}&uri={uri}")]
        Student RegisterPushNotificationChannelForStudent(string studentCode, string uri);

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle =
WebMessageBodyStyle.Bare, UriTemplate =
"RegisterStudentForFinalExam/{studentCode}/{courseCode}")]
        TaskSummary RegisterStudentForFinalExam(string studentCode, string courseCode);
    }
}
```

Service1.svc.cs

```
using System;
using System.Web;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Data.Helpers;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using CareerDto = TesinaMobileCloud.Data.DTO.Career;
using Course = TesinaMobileCloud.Data.DTO.Course;
using Student = TesinaMobileCloud.Data.DTO.Student;

namespace SiaService
{
    public class SiaService : ISiaService
    {
        private readonly ICareerRepository _careerRepository;
        private readonly ICourseRepository _courseRepository;
```

```
        private readonly ICourseCareerRelationshipRepository
_courseCareerRelationshipRepository;
        private readonly IStudentRepository _studentRepository;
        private readonly IStudentCourseRepository _studentCourseRelationshipRepository;

        public SiaService(ICourseRepository courseRepository,
ICourseCareerRelationshipRepository courseCareerRelationshipRepository,
IStudentRepository studentRepository, IStudentCourseRepository
studentCourseRelationshipRepository, ICareerRepository careerRepository)
        {
            _courseRepository = courseRepository;
            _courseCareerRelationshipRepository = courseCareerRelationshipRepository;
            _studentRepository = studentRepository;
            _studentCourseRelationshipRepository = studentCourseRelationshipRepository;
            _careerRepository = careerRepository;
        }

        public Student RetriveStudentByCodeAndCareer(string studentCode, string
careerCode)
        {
            var studentAzure =
_studentRepository.GetStudentByCareerCodeAndStudentCode(studentCode, careerCode);
            var studentToRetrive = new Student
                {
                    FirstName = studentAzure.FirstName,
                    LastName = studentAzure.LastName,
                    StudentCode = studentAzure.StudentCode,
                    CareerCode = studentAzure.CareerCode,
                    DeviceUri = studentAzure.DeviceUri
                };
            var coursesOfStudent =
_studentCourseRelationshipRepository.RetriveCoursesOfStudent(studentCode);
            foreach (var studentCourseRelationship in coursesOfStudent)
            {
                var singleCourse =
_courseRepository.GetSingleCourse(studentCourseRelationship.CourseCode);
                studentToRetrive.Courses.Add(BuildStudentCourseDto(careerCode,
studentCourseRelationship, singleCourse));
            }

            var career = _careerRepository.GetCareer(careerCode);
            studentToRetrive.Career = new CareerDto
                {
                    Code = career.CareerCode.ToString(),
                    Title = career.Title
                };
```

```
            return studentToRetrive;
        }

        private StudentCourse BuildStudentCourseDto(string careerCode,
StudentCourseRelationship studentCourseRelationship,
TesinaMobileCloud.Data.Model.Course singleCourse)
        {
            return new StudentCourse
                {
                    FinalExamResult = studentCourseRelationship.FinalExamResult,
                    PartialExamResult = studentCourseRelationship.PartialExamResult,
                    PracticWorkResult = studentCourseRelationship.PracticWorkResult,
                    RecuperationExamResult =
studentCourseRelationship.RecuperationExamResult,
                    TotalHoursAssisted = studentCourseRelationship.TotalHoursAssisted,
                    FinalExamInscripted = studentCourseRelationship.FinalExamInscripted,
                    IsCurretCourse = studentCourseRelationship.IsCurrentCourse,
                    AttendanceState =
GetIntFromAttendanceState(studentCourseRelationship.GetAttendanceState(singleCourse.
TotalHoursDictated, singleCourse.IsCritical)),
                    Course = new Course
                            {
                                Code = singleCourse.Code,
                                Professor = singleCourse.Professor,
                                Title = singleCourse.Title,
                                FinalExamDate = singleCourse.FinalExamDate,
                                PartialExamDate = singleCourse.PartialExameDate,
                                PracticWorkPresentationDate =
singleCourse.PracticWorkPresentationDate,
                                RecuperationExameDate = singleCourse.RecuperationExameDate,
                                Scheduled = singleCourse.Scheduled,
                                Description = singleCourse.Description,
                                YearOfCareer =
_courseCareerRelationshipRepository.GetCareerYearOfCourseByCareer(int.Parse(career
Code), singleCourse.Code),
                                TotalHoursOfClasses = singleCourse.TotalHoursOfClasses,
                                TotalHoursInAWeek = singleCourse.TotalHoursInAWeek,
                                TotalHoursDictated = singleCourse.TotalHoursDictated
                            }
                };
        }

        private int GetIntFromAttendanceState(AttendanceState attendanceState)
        {
            switch (attendanceState)
            {
                case AttendanceState.Danger:
```

```
                return 1;
            case AttendanceState.Precaution:
                return 2;
            default:
                return 3;
        }
    }

    public StudentCourse RetriveStudentCourse(string careerCode, string studentCode,
string courseCode)
    {
        var studentCourse =
_studentCourseRelationshipRepository.RetriveStudentCourseByCode(studentCode,
courseCode);
        var course = _courseRepository.GetSingleCourse(studentCourse.CourseCode);

        return BuildStudentCourseDto(careerCode, studentCourse, course);
    }

    public Student RegisterPushNotificationChannelForStudent(string studentCode, string
uri)
    {
        var student = _studentRepository.GetStudentByStudentCode(studentCode);
        student.DeviceUri = HttpUtility.UrlDecode(uri);
        _studentRepository.Add(student);

        return new Student();
    }


    public TaskSummary RegisterStudentForFinalExam(string studentCode, string
courseCode)
    {
        var registerStudentForFinalExamSuccess = new TaskSummary { Result =
TaskResult.Failed, Message = "La inscripción se ha realizado con exito." };
        var registerStudentForFinalExamFail = new TaskSummary { Result =
TaskResult.Failed, Message = "No se puede realizar la inscripción. No se cumplen los
requisitos necesarios." };

        try
        {
            var studentCourse =
_studentCourseRelationshipRepository.RetriveStudentCourseByCode(studentCode,
courseCode);
            var course =
_courseRepository.GetSingleCourse(Convert.ToInt32(courseCode));
```

```
                if (studentCourse.FinalExamInscripted)
                {
                    return registerStudentForFinalExamFail;
                }
                if (studentCourse.GetAttendanceState(course.TotalHoursDictated,
            course.IsCritical) != AttendanceState.Danger)
                {
                    if (studentCourse.FinalExamResult == 0)
                        if (studentCourse.PartialExamResult >= 4 ||
            studentCourse.RecuperationExamResult >= 4)
                        {
                            if (!studentCourse.FinalExamInscripted)
                            {
                                studentCourse.FinalExamInscripted = true;
                                return registerStudentForFinalExamSuccess;
                            }
                            return registerStudentForFinalExamFail;

                        }
                        else
                            return registerStudentForFinalExamFail;
                    return registerStudentForFinalExamFail;
                }
                return registerStudentForFinalExamFail;


            }
            catch (Exception)
            {
                throw;
            }
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace SiaService
{
    public class WebRole : RoleEntryPoint
```

```
    {
        public override bool OnStart()
        {
            // To enable the AzureLocalStorageTraceListner, uncomment relevent section in the
web.config
            DiagnosticMonitorConfiguration diagnosticConfig =
DiagnosticMonitor.GetDefaultInitialConfiguration();
            diagnosticConfig.Directories.ScheduledTransferPeriod =
TimeSpan.FromMinutes(1);

diagnosticConfig.Directories.DataSources.Add(AzureLocalStorageTraceListener.GetLogDir
ectory());

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

TesinaMobileCloud.AdminSite

Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
```

```
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
    }
}
```

WebRole.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.AdminSite
{
    public class WebRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

            return base.OnStart();
        }
    }
}
```

BundleConfig.cs

```
using System.Web;
using System.Web.Optimization;

namespace TesinaMobileCloud.AdminSite
{
    public class BundleConfig
    {
        // For more information on Bundling, visit
http://go.microsoft.com/fwlink/?LinkId=254725
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                        "~/Scripts/jquery-ui-{version}.js"));
```

```
            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then,
        when you're
            // ready for production, use the build tool at http://modernizr.com to pick only the
        tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

            bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
                        "~/Content/themes/base/jquery.ui.core.css",
                        "~/Content/themes/base/jquery.ui.resizable.css",
                        "~/Content/themes/base/jquery.ui.selectable.css",
                        "~/Content/themes/base/jquery.ui.accordion.css",
                        "~/Content/themes/base/jquery.ui.autocomplete.css",
                        "~/Content/themes/base/jquery.ui.button.css",
                        "~/Content/themes/base/jquery.ui.dialog.css",
                        "~/Content/themes/base/jquery.ui.slider.css",
                        "~/Content/themes/base/jquery.ui.tabs.css",
                        "~/Content/themes/base/jquery.ui.datepicker.css",
                        "~/Content/themes/base/jquery.ui.progressbar.css",
                        "~/Content/themes/base/jquery.ui.theme.css"));
        }
    }
}
```

FilterConfig.cs

```
using System.Web;
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

NinjectWebCommon.cs

```
using Ninject.Modules;
```

```
using TesinaMobileCloud.Shared;

[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NinjectWebCommon), "Start")]
[assembly:
WebActivator.ApplicationShutdownMethodAttribute(typeof(TesinaMobileCloud.AdminSite.A
pp_Start.NinjectWebCommon), "Stop")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;

    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
```

```
            var modules = new INinjectModule[] { new ServiceModule() };
            var kernel = new StandardKernel(modules);
            kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new Bootstrapper().Kernel);
            kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

            RegisterServices(kernel);
            return kernel;
        }

        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
}
```

NotificationManagementUi.cs

```
[assembly:
WebActivator.PreApplicationStartMethod(typeof(TesinaMobileCloud.AdminSite.App_Start.
NotificationManagementUi), "PreStart")]

namespace TesinaMobileCloud.AdminSite.App_Start
{
    using System.Reflection;
    using Microsoft.WindowsAzure;
    using Microsoft.WindowsAzure.StorageClient;
    using TesinaMobileCloud.AdminSite.Areas.Notifications.Helpers;

    public static class NotificationManagementUi
    {
        public const string TileImagesContainerName = "tileimagescontainer";

        public static void PreStart()
        {
            NotificationsManagementUiContext.Configure = c =>
                {
                    // TODO: Replace with your own Windows Azure Storage account name and
key, or read it from a configuration file
                    c.CloudStorageAccount = CloudStorageAccount.DevelopmentStorageAccount;

                    // TODO: Replace with your prefered container name to store tile images
                    c.TileImagesContainerName = TileImagesContainerName;
```

```
                return c;
            };


        UploadTileImage("TesinaMobileCloud.AdminSite.Resources.WindowsAzureLogo.png");

        UploadTileImage("TesinaMobileCloud.AdminSite.Resources.WindowsPhoneLogo.png");

        UploadTileImage("TesinaMobileCloud.AdminSite.Resources.AzureBackground.png");

        UploadTileImage("TesinaMobileCloud.AdminSite.Resources.DefaultBackground.png");
        }

        private static void UploadTileImage(string imageName)
        {
            var cloudBlobClient =
NotificationsManagementUiContext.Current.CloudStorageAccount.CreateCloudBlobClient()
;
            var tileImagesContainerName =
NotificationsManagementUiContext.Current.TileImagesContainerName;

            // Create the container (and make it public)
            var container = cloudBlobClient.GetContainerReference(tileImagesContainerName);
            container.CreateIfNotExist();
            container.SetPermissions(
                new BlobContainerPermissions
                {
                    PublicAccess = BlobContainerPublicAccessType.Blob
                });

            // Upload the image from the assembly resources.
            var assembly = Assembly.GetExecutingAssembly();
            var imageStream = assembly.GetManifestResourceStream(imageName);
            var blob =
container.GetBlobReference(imageName.Replace("TesinaMobileCloud.AdminSite.Resourc
es.", string.Empty));
            blob.Properties.ContentType = "image/png";
            blob.UploadFromStream(imageStream);
        }
    }
}
```

RouteConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.Mvc;
using System.Web.Routing;

namespace TesinaMobileCloud.AdminSite
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

WebApiConfig.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace TesinaMobileCloud.AdminSite
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

Controllers

CareerController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerController : Controller
    {
        private readonly ITable<Career> _careers;

        public CareerController(ITable<Career> careers)
        {
            _careers = careers;
        }

//
        // GET: /Career/

        public ActionResult Index()
        {
            return View(_careers.GetByCriteria(career =>
!string.IsNullOrEmpty(career.PartitionKey)));
        }

        //
        // GET: /Career/Details/5

        public ActionResult Details(int id)
        {
            return View(_careers.GetSingleByCriteria(career => career.CareerCode == id));
        }

        //
        // GET: /Career/Create

        public ActionResult Create()
        {
            return View(new Career());
        }

        //
        // POST: /Career/Create

        [HttpPost]
        public ActionResult Create(Career career)
        {
```

```
            try
            {
                _careers.Add(career);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_careers.GetSingleByCriteria(career1 => career1.CareerCode == id));
        }

        //
        // POST: /Career/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var career = _careers.GetSingleByCriteria(career1 => career1.CareerCode ==
id);
                UpdateModel(career);
                _careers.Add(career);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Career/Delete/5

        public ActionResult Delete(int id)
        {
            return View();
        }
```

```
        //
        // POST: /Career/Delete/5

        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                _careers.Delete(_careers.GetSingleByCriteria(c => c.CareerCode == id));

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

CareerCourseRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CareerCourseRelationshipController : Controller
    {
        private readonly ITable<CareerCourseRelationship> _careerCourseRelationship;

        public CareerCourseRelationshipController(ITable<CareerCourseRelationship>
careerCourseRelationship)
        {
            _careerCourseRelationship = careerCourseRelationship;
        }

//
        // GET: /CareerCourseRelationship/

        public ActionResult Index()
```

```
        {
            return View(_careerCourseRelationship.GetByCriteria(relationship =>
relationship.CareerCode != 0 ));
        }

        //
        // GET: /CareerCourseRelationship/Details/5

        public ActionResult Details(int course, int career)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CourseCode == course && relationship.CareerCode == career));
        }

        //
        // GET: /CareerCourseRelationship/Create

        public ActionResult Create()
        {
            return View(new CareerCourseRelationship());
        }

        //
        // POST: /CareerCourseRelationship/Create

        [HttpPost]
        public ActionResult Create(CareerCourseRelationship careerCourseRelationship)
        {
            try
            {
                _careerCourseRelationship.Add(careerCourseRelationship);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /CareerCourseRelationship/Edit/5

        public ActionResult Edit(int course, int career)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(relationship =>
relationship.CourseCode == course && relationship.CareerCode == career));
        }
```

```
        //
        // POST: /CareerCourseRelationship/Edit/5

        [HttpPost]
        public ActionResult Edit(int careerCode, int courseCode, FormCollection collection)
        {
            try
            {
                var courseInCloud = _careerCourseRelationship.GetSingleByCriteria(s =>
s.CareerCode == careerCode && s.CourseCode == courseCode);
                UpdateModel(courseInCloud);
                _careerCourseRelationship.Add(courseInCloud);

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /CareerCourseRelationship/Delete/5

        public ActionResult Delete(int careerCode, int courseCode)
        {
            return View(_careerCourseRelationship.GetSingleByCriteria(s => s.CareerCode ==
careerCode && s.CourseCode == courseCode));
        }

        //
        // POST: /CareerCourseRelationship/Delete/5

        [HttpPost]
        public ActionResult Delete(int careerCode, int courseCode, FormCollection collection)
        {
            try
            {

_careerCourseRelationship.Delete(_careerCourseRelationship.GetSingleByCriteria(c =>
c.CareerCode == careerCode && c.CourseCode == courseCode));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
```

```
            }
        }
    }
}
```

CourseController.cs

```
using System;
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class CourseController : Controller
    {
        private readonly ITable<Course> _courses;

        public CourseController(ITable<Course> courses)
        {
            _courses = courses;
        }

        public ActionResult Index()
        {
            return View(_courses.GetByCriteria(course =>
!string.IsNullOrEmpty(course.PartitionKey)));
        }

        //
        // GET: /Course/Details/5

        public ActionResult Details(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.Code == id));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new Course(0, string.Empty));
        }

        //
        // POST: /Course/Create
```

```
        [HttpPost]
        public ActionResult Create(Course course)
        {
            try
            {
                _courses.Add(course);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id)
        {
            return View(_courses.GetSingleByCriteria(course => course.Code == id));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _courses.GetSingleByCriteria(course => course.Code ==
id);
                UpdateModel(courseInCloud);
                _courses.Add(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id)
```

```
        {
            try
            {
                _courses.Delete( _courses.GetSingleByCriteria(c => c.PartitionKey == id));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // POST: /Course/Delete/5


    }
}
```

HomeController.cs

```
using System.Web.Mvc;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Sitio de administración y soporte, Tesina Carlos Rodrigo";

            return View();
        }
    }
}
```

StudentController.cs

```
using System.Web.Mvc;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class StudentController : Controller
    {
        private readonly ITable<Student> _students;
```

```
public StudentController(ITable<Student> students)
{
    _students = students;
}

public ActionResult Index()
{
    return View(_students.GetByCriteria(s => !string.IsNullOrEmpty(s.PartitionKey)));
}

public ActionResult Details(int id)
{
    return View(_students.GetSingleByCriteria(s => s.StudentCode == id));
}

//
// GET: /Course/Create

public ActionResult Create()
{
    return View(new Student());
}

//
// POST: /Course/Create

[HttpPost]
public ActionResult Create(Student student)
{
    try
    {
        _students.Add(student);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /Course/Edit/5

public ActionResult Edit(int id)
{
    return View(_students.GetSingleByCriteria(s => s.StudentCode == id));
```

```
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var courseInCloud = _students.GetSingleByCriteria(s => s.StudentCode == id);
                UpdateModel(courseInCloud);
                _students.Add(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
                _students.Delete(_students.GetSingleByCriteria(c => c.PartitionKey == id &&
c.RowKey == row));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

StudentCourseRelationshipController.cs

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Web;
using System.Web.Helpers;
```

```
using System.Web.Mvc;
using Newtonsoft.Json;
using TesinaMobileCloud.Data.Helpers;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Data.Table;
using TesinaMobileCloud.Notifications;

namespace TesinaMobileCloud.AdminSite.Controllers
{
    public class StudentCourseRelationshipController : Controller
    {
        private readonly ITable<StudentCourseRelationship> _table;
        private readonly ITable<Course> _courses;
        private readonly ITable<Student> _students;
        private readonly IQueue<QueueNotificationMessage> _queueNotifications;

        public StudentCourseRelationshipController(ITable<StudentCourseRelationship>
table, ITable<Course> courses, IQueue<QueueNotificationMessage> queueNotifications,
ITable<Student> students)
        {
            _table = table;
            _courses = courses;
            _queueNotifications = queueNotifications;
            _students = students;
        }

        public ActionResult Index()
        {
            return View(_table.GetByCriteria(relationship => relationship.StudentCode != 0));
        }

        //
        // GET: /StudentCodeRelationship/Details/5

        public ActionResult Details(int id, int courseCode)
        {
            return View(_table.GetSingleByCriteria(s => s.StudentCode == id && s.CourseCode
== courseCode));
        }

        //
        // GET: /Course/Create

        public ActionResult Create()
        {
            return View(new StudentCourseRelationship());
```

```
        }

        //
        // POST: /Course/Create

        [HttpPost]
        public ActionResult Create(StudentCourseRelationship student)
        {
            try
            {
                _table.Add(student);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Course/Edit/5

        public ActionResult Edit(int id, int courseCode)
        {
            return View(_table.GetSingleByCriteria(s => s.StudentCode == id && s.CourseCode
== courseCode));
        }

        //
        // POST: /Course/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, int courseCode, FormCollection collection)
        {
            try
            {
                var courseInCloud = _table.GetSingleByCriteria(s => s.StudentCode == id &&
s.CourseCode == courseCode);
                UpdateModel(courseInCloud);
                _table.Add(courseInCloud);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
```

```
        //
        // GET: /Course/Delete/5

        public ActionResult Delete(string id, string row)
        {
            try
            {
                _table.Delete(_table.GetSingleByCriteria(c => c.PartitionKey == id && c.RowKey
== row));
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        public ActionResult ExamResultAsign(int id, int course)
        {
            try
            {
                ViewBag.ExamTypes =
                    new List<SelectListItem>
                    {
                        new SelectListItem {Text = "Parcial", Value = "PartialExam"},
                        new SelectListItem {Text = "Recuperatorio", Value =
"RecuperationExam"},
                        new SelectListItem {Text = "Final", Value = "FinalExam"},
                        new SelectListItem
                        {
                            Text = "Presentación de Trabajos Practicos",
                            Value = "PracticalWorkExam"
                        },
                    };

                return View(_table.GetSingleByCriteria(s => s.StudentCode == id &&
s.CourseCode == course));
            }
            catch
            {
                return View();
            }
        }

        [HttpPost]
```

```
        public ActionResult ExamResultAsign(int studentCode, int courseCode,
FormCollection form)
        {
            var selectListItems = new List<SelectListItem>
                {
                    new SelectListItem {Text = "Parcial", Value = "PartialExam"},
                    new SelectListItem {Text = "Recuperatorio", Value = "RecuperationExam"},
                    new SelectListItem {Text = "Final", Value = "FinalExam"},
                    new SelectListItem {Text = "Presentación de Trabajos Practicos", Value =
"PracticalWorkExam"},
                };
            try
            {
                var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode ==
studentCode && sc.CourseCode == courseCode);
                var course = _courses.GetSingleByCriteria(c => c.Code == courseCode);
                var student = _students.GetSingleByCriteria(s => s.StudentCode ==
studentCode);
                var examType = form["SelectedExamType"];
                var examResult = int.Parse(form["ExamResultText"]);

                if (examResult > 10 || examResult == 0)
                {
                    ModelState.AddModelError(string.Empty, "Calificación no valida");
                    ViewBag.ExamTypes =
                     selectListItems;
                    return View(_table.GetSingleByCriteria(s => s.StudentCode == studentCode
&& s.CourseCode == courseCode));
                }



                var tileNotification = new TileNotification
                {
                    BackBackgroundImage = "Void",
                    BackTitle = " "
                };

                var toastNotification = new ToastNotification
                {
                    Title = course.Title,
                    Page =
String.Format("/View/CourseView.xaml?CareerCode={0}&amp;StudentCode={1}&amp;Cou
rseCode={2}",
                                  student.CareerCode,
                                  student.StudentCode,
                                  courseCode)
```

```
            };

            switch (examType)
            {
                case "PartialExam":
                    studentCourse.PartialExamResult = examResult;
                    var partialExamResult = String.Format("Examen Parcial: {0}", examResult);
                    tileNotification.BackContent = course.Title + ". " + partialExamResult;
                    toastNotification.Content = partialExamResult;
                    break;
                case "RecuperationExam":
                    studentCourse.RecuperationExamResult = examResult;
                    var recuperationExamResult = String.Format("Examen Recuperatorio: {0}",
examResult);
                    tileNotification.BackContent = course.Title + ". " + recuperationExamResult; ;
                    toastNotification.Content = recuperationExamResult;
                    break;
                case "FinalExam":
                    studentCourse.FinalExamResult = examResult;
                    var finalExamResult = String.Format("Examen Final: {0}", examResult);
                    tileNotification.BackContent = course.Title + ". " + finalExamResult; ;
                    toastNotification.Content = finalExamResult;
                    break;
                default:
                    studentCourse.PracticWorkResult = examResult;
                    var tpResult = String.Format("Presentación TPs: {0}", examResult);
                    tileNotification.BackContent = course.Title + ". " + tpResult; ;
                    toastNotification.Content = tpResult;
                    break;
            }

            _table.Add(studentCourse);

            _queueNotifications.AddMessage(new QueueNotificationMessage
            {
                Destinatary = student.DeviceUri,
                Type = tileNotification.Type,
                Payload = tileNotification.Payload
            });

            _queueNotifications.AddMessage(new QueueNotificationMessage
            {
                Destinatary = student.DeviceUri,
                Type = toastNotification.Type,
                Payload = toastNotification.Payload
            });
```

```
            return RedirectToAction("Index");
        }
        catch (Exception)
        {
            ViewBag.ExamTypes =
                selectListItems;
            return View(_table.GetSingleByCriteria(s => s.StudentCode == studentCode &&
s.CourseCode == courseCode));
        }
    }

    public ActionResult AsignAssistedHours(int id, int course)
    {
        var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode == id &&
sc.CourseCode == course);
        return View(studentCourse);
    }

    [HttpPost]
    public ActionResult AsignAssistedHours(int studentCode, int courseCode,
FormCollection collection)
    {
        var studentCourse = _table.GetSingleByCriteria(sc => sc.StudentCode ==
studentCode && sc.CourseCode == courseCode);
        var course = _courses.GetSingleByCriteria(c => c.Code == courseCode);
        var studentDeviceUri = _students.GetSingleByCriteria(s => s.StudentCode ==
studentCode).DeviceUri;

        var previousAssistanceState =
studentCourse.GetAttendanceState(course.TotalHoursDictated, course.IsCritical);
        try
        {
            var hours = int.Parse(collection["AddHours"]);

            studentCourse.TotalHoursAssisted += hours;
            _table.Add(studentCourse);

            AddAttedanceChangeNotification(studentCourse, course, studentDeviceUri,
previousAssistanceState);

            return RedirectToAction("Index");
        }
        catch (Exception)
        {
            return View(studentCourse);
        }
    }
```

```
        private void AddAttedanceChangeNotification(StudentCourseRelationship
studentCourse, Course course, string studentDeviceUri, AttendanceState
previousAssistanceState)
        {
            if (!string.IsNullOrEmpty(studentDeviceUri))
            {
                var actualAssistanceState =
studentCourse.GetAttendanceState(course.TotalHoursDictated, course.IsCritical);
                if (previousAssistanceState != actualAssistanceState)
                {
                    var tileNotification = new TileNotification
                    {
                        BackTitle = string.Format("Asist: {0}",
GetActualAssistanceStateText(actualAssistanceState)),
                        BackContent = course.Title,
                        BackBackgroundImage = actualAssistanceState.ToString()
                    };

                    _queueNotifications.AddMessage(new QueueNotificationMessage
                    {
                        Destinatary = studentDeviceUri,
                        Type = tileNotification.Type,
                        Payload = tileNotification.Payload
                    });
                }
            }
        }

        private static string GetActualAssistanceStateText(AttendanceState
actualAssistanceState)
        {
            switch (actualAssistanceState)
            {
                case AttendanceState.Danger:
                    return "Baja";
                case AttendanceState.Precaution:
                    return "Cuidado";
                default:
                    return "Buena";
            }
        }
    }
}
```

Views

_ViewStart.cshtml

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Career

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Nueva";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Dar de Alta Carrera</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
```

```
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Eliminar</h2>

<h3>Está seguro de eliminar esta Carrera?</h3>
<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
@using (Html.BeginForm())
{
    <p>
        <input type="submit" value="Eliminar" />
        |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Details</h2>

<fieldset>
    <legend>Career</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { id=Model.CareerCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Career

@{
    ViewBag.Title = "Editar";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
```

```
        <legend>Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Career>

@{
    ViewBag.Title = "Carreras";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Carreras</h2>

<p>
    @Html.ActionLink("Nueva", "Create")
</p>
<table>
```

```
    <tr>
        <th>
            @Html.Label("CareerCode", "Código")
        </th>
        <th>
            @Html.Label("Title", "Título")
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { id = item.CareerCode }) |
            @Html.ActionLink("Detalles", "Details", new { id = item.CareerCode }) |
            @Html.ActionLink("Eliminar", "Delete", new { id = item.CareerCode })
        </td>
    </tr>
}

</table>
```

CareerCourseRelationship

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Asignar Materia a Carrera</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>
```

```
        <div class="editor-label">
            @Html.Label("CareerCode", "Código de la Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de la Materia")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("YearInTheCareer", "Año de Cursada")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>


        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Delete";
}
```

```
<h2>Eliminar Relación</h2>

<h3>Está seguro que desea eliminar esta relación?</h3>
<fieldset>
    <legend>Relación Materia-Carrera</legend>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Materia")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("YearInTheCareer", "Año en que se dicta")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.YearInTheCareer)
    </div>

</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Relación Materia-Carrera</legend>
```

```
    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Materia")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("YearInTheCareer", "Año en que se dicta")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.YearInTheCareer)
    </div>

</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { course = Model.CourseCode, career =
Model.CareerCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.CareerCourseRelationship

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Relación Materia-Carrera</legend>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
```

```
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Materia")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("YearInTheCareer", "Año en que se dicta")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.YearInTheCareer)
            @Html.ValidationMessageFor(model => model.YearInTheCareer)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.CareerCourseRelationship>

@{
    ViewBag.Title = "Asignar Materia a Carrera";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Asignación de Materias a Carreras</h2>
```

```
<p>
   @Html.ActionLink("Nueva asignación", "Create")
</p>
<table>
   <tr>
      <th>
         @Html.Label("CareerCode", "Código de Carrera")
      </th>
      <th>
         @Html.Label("CourseCode", "Código de Materia")
      </th>
      <th>
         @Html.Label("YearInTheCareer", "Año en que se dicta")
      </th>

      <th></th>
   </tr>

@foreach (var item in Model) {
   <tr>
      <td>
         @Html.DisplayFor(modelItem => item.CareerCode)
      </td>
      <td>
         @Html.DisplayFor(modelItem => item.CourseCode)
      </td>
      <td>
         @Html.DisplayFor(modelItem => item.YearInTheCareer)
      </td>
      <td>
         @Html.ActionLink("Editar", "Edit", new     { course = item.CourseCode, career =
item.CareerCode }) |
         @Html.ActionLink("Detalles", "Details",new { course = item.CourseCode, career =
item.CareerCode }) |
         @Html.ActionLink("Eliminar", "Delete", new { courseCode = item.CourseCode,
careerCode = item.CareerCode } )
      </td>
   </tr>
}

</table>
```

Course

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Course
```

```
@{
    ViewBag.Title = "Nuevo";
}

<h2>Materia</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Materia</legend>

        <div class="editor-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Code)
            @Html.ValidationMessageFor(model => model.Code)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.Label("Professor", "Docente")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>
        <div class="editor-label">
            @Html.Label("FinalExamDate", "Fecha de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamDate)
            @Html.ValidationMessageFor(model => model.FinalExamDate)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
        </div>
```

```
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExameDate)
            @Html.ValidationMessageFor(model => model.PartialExameDate)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkPresentationDate)
            @Html.ValidationMessageFor(model => model.PracticWorkPresentationDate)
        </div>
        <div class="editor-label">
            @Html.Label("RecuperationExameDate", "Fecha de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExameDate)
            @Html.ValidationMessageFor(model => model.RecuperationExameDate)
        </div>
        <div class="editor-label">
             @Html.Label("TotalHoursOfClasses", "Total de horas catedra de la materia")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-label">
             @Html.Label("TotalHoursInAWeek", "Total de horas catedra semanales")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursInAWeek)
            @Html.ValidationMessageFor(model => model.TotalHoursInAWeek)
        </div>
        <div class="editor-label">
             @Html.Label("TotalHoursDictated", "Total de horas catedra dicatadas")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursDictated)
            @Html.ValidationMessageFor(model => model.TotalHoursDictated)
        </div>
        <div class="editor-label">
            @Html.Label("Scheduled", "Horario")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Scheduled)
            @Html.ValidationMessageFor(model => model.Scheduled)
```

```
        </div>

        <div class="editor-label">
            @Html.Label("IsCritical", "Es Troncal")
        </div>
        <div class="editor-field">
            @Html.CheckBoxFor(model => model.IsCritical)
            @Html.ValidationMessageFor(model => model.IsCritical)
        </div>

        <div class="editor-label">
            @Html.Label("Description", "Descripción de la Materia")
        </div>

        <div class="editor-field">
            @Html.TextAreaFor(model => model.Description)
            @Html.ValidationMessageFor(model => model.Description)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar</h2>

<h3>Está seguro que desea eliminar la Materia?</h3>
<fieldset>
    <legend>Course</legend>

    <div class="display-label">
```

```
            @Html.Label("Code", "Código")
        </div>
        <div class="display-field">
            @Html.DisplayFor(model => model.Code)
        </div>

        <div class="display-label">
            @Html.Label("Title", "Título")
        </div>
        <div class="display-field">
            @Html.DisplayFor(model => model.Title)
        </div>

        <div class="display-label">
            @Html.Label("Professor", "Docente")
        </div>
        <div class="display-field">
            @Html.DisplayFor(model => model.Professor)
        </div>
    </fieldset>
    @using (Html.BeginForm()) {
        <p>
            <input type="submit" value="Eliminar" /> |
            @Html.ActionLink("Volver", "Index")
        </p>
    }
```

Details.cshtml

```
    @model TesinaMobileCloud.Data.Model.Course

    @{
        ViewBag.Title = "Detalles";
    }

    <h2>Detalles</h2>

    <fieldset>
        <legend>Materia</legend>

        <div class="display-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="display-field">
            @Html.DisplayFor(model => model.Code)
        </div>
```

```
    <div class="display-label">
        @Html.Label("Title", "Título")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Title)
    </div>

    <div class="display-label">
        @Html.Label("Professor", "Docente")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.Professor)
    </div>
    <div class="editor-label">
        @Html.Label("FinalExamDate", "Fecha de Examen Final")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.FinalExamDate)
    </div>

    <div class="editor-label">
        @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.PartialExameDate)
    </div>

    <div class="editor-label">
        @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.PracticWorkPresentationDate)
    </div>
    <div class="editor-label">
        @Html.Label("RecuperationExame", "Fecha de Examen Recuperatorio")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.RecuperationExameDate)
    </div>
    <div class="editor-label">
        @Html.Label("TotalHoursOfClasses", "Total de Horas de Cursada")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.TotalHoursOfClasses)
    </div>
```

```
    <div class="editor-label">
        @Html.Label("Scheduled", "Horario")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.Scheduled)
    </div>

    <div class="editor-label">
        @Html.Label("Description", "Descripción de la Materia")
    </div>
    <div class="editor-field">
        @Html.DisplayFor(model => model.Description)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new { id = Model.Code }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Course

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Materia</legend>

        <div class="editor-label">
            @Html.Label("Code", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Code)
            @Html.ValidationMessageFor(model => model.Code)
        </div>

        <div class="editor-label">
            @Html.Label("Title", "Título")
        </div>
```

```
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>

        <div class="editor-label">
            @Html.Label("Professor", "Docente")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Professor)
            @Html.ValidationMessageFor(model => model.Professor)
        </div>

       <div class="editor-label">
            @Html.Label("FinalExamDate", "Fecha de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamDate)
            @Html.ValidationMessageFor(model => model.FinalExamDate)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExameDate", "Fecha de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExameDate)
            @Html.ValidationMessageFor(model => model.PartialExameDate)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkPresentationDate", "Fecha de Presentación de Trabajos
Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkPresentationDate)
            @Html.ValidationMessageFor(model => model.PracticWorkPresentationDate)
        </div>
        <div class="editor-label">
            @Html.Label("RecuperationExameDate", "Fecha de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExameDate)
            @Html.ValidationMessageFor(model => model.RecuperationExameDate)
        </div>
         <div class="editor-label">
             @Html.Label("TotalHoursOfClasses", "Total de horas catedra de la materia")
        </div>
```

```
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursOfClasses)
            @Html.ValidationMessageFor(model => model.TotalHoursOfClasses)
        </div>
        <div class="editor-label">
             @Html.Label("TotalHoursInAWeek", "Total de horas catedra semanales")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursInAWeek)
            @Html.ValidationMessageFor(model => model.TotalHoursInAWeek)
        </div>
        <div class="editor-label">
             @Html.Label("TotalHoursDictated", "Total de horas catedra dicatadas")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursDictated)
            @Html.ValidationMessageFor(model => model.TotalHoursDictated)
        </div>
        <div class="editor-label">
            @Html.Label("Scheduled", "Horario")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Scheduled)
            @Html.ValidationMessageFor(model => model.Scheduled)
        </div>

        <div class="editor-label">
            @Html.Label("IsCritical", "Es Troncal")
        </div>
        <div class="editor-field">
            @Html.CheckBoxFor(model => model.IsCritical)
            @Html.ValidationMessageFor(model => model.IsCritical)
        </div>

        <div class="editor-label">
            @Html.Label("Description", "Descripción de la Materia")
        </div>
        <div class="editor-field">
            @Html.TextAreaFor(model => model.Description)
            @Html.ValidationMessageFor(model => model.Description)
        </div>

        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}
```

```
<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Course>

@{
    ViewBag.Title = "Materias";
}
<style type="text/css">
th {
    padding-right: 10px;
}
</style>
<h2>Materias</h2>

<p>
    @Html.ActionLink("Nueva", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("Code", "Código")
        </th>
        <th>
            @Html.Label("Title", "Título")
        </th>
        <th>
            @Html.Label("Professor", "Docente")
        </th>
        <th>
            @Html.Label("Scheduled", "Horario")
        </th>
        <th>
            @Html.Label("TotalHoursOfClasses", "Horas Totales de Cursada")
        </th>
        <th>
            @Html.Label("TotalHoursInAWeek", "Horas Semanales de Cursada")
        </th>
        <th>
```

```
            @Html.Label("TotalHoursDictated", "Horas Dictadas")
        </th>
        <th>
            @Html.Label("IsCritical", "Es Troncal")
        </th>
        <th></th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Code)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Professor)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Scheduled)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursOfClasses) horas catedra
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursInAWeek) horas catedra
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursDictated) horas catedra
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.IsCritical)
            </td>
            <td>
                @Html.ActionLink("Editar", "Edit", new { id = item.Code  }) |
            @Html.ActionLink("Detalles", "Details", new { id = item.Code }) |
            @Html.ActionLink("Eliminar", "Delete", new { id = item.Code })
            </td>
        </tr>
    }

</table>
```

Home

Index.cshtml

```
@{
    ViewBag.Title = "Sitio de Administración - Tesina Carlos Rodrigo";
}
@section featured {
    <section class="featured">
        <div class="content-wrapper">
            <hgroup class="title">
                <h1>@ViewBag.Title.</h1>
            </hgroup>
            <p>
                Este sitio provee las funciones de administración y carga de datos necesarios
para el desarrollo de la tesina.
                La necesidad surge de simular los sistemas de asistencias y administración de
los alumnos, para solo dedicarse a
                la implementación del cliente Mobile.
            </p>
        </div>
    </section>
}
```

Shared

Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<hgroup class="title">
    <h1 class="error">Error.</h1>
    <h2 class="error">Se ha producido un error inesperado.</h2>
</hgroup>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width" />
```

```
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p class="site-title">@Html.ActionLink("your logo here", "Index", "Home")</p>
                </div>
                <div class="float-right">
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("Materias", "Index", "Course")</li>
                            <li>@Html.ActionLink("Carreras", "Index", "Career")</li>
                            <li>@Html.ActionLink("Materias-Carreras", "Index",
"CareerCourseRelationship")</li>
                            <li>@Html.ActionLink("Estudiantes", "Index", "Student")</li>
                            <li>@Html.ActionLink("Estudiantes-Materias", "Index",
"StudentCourseRelationship")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
            </div>
        </footer>

        @Scripts.Render("~/bundles/jquery")
        @RenderSection("scripts", required: false)
    </body>
</html>
```

_LoginPartial.cshtml

```
@if (Request.IsAuthenticated) {
```

```
    <text>
        Hello, @Html.ActionLink(User.Identity.Name, "Manage", "Account", routeValues: null,
htmlAttributes: new { @class = "username", title = "Manage" })!
        @using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id =
"logoutForm" })) {
            @Html.AntiForgeryToken()
            <a href="javascript:document.getElementById('logoutForm').submit()">Log off</a>
        }
    </text>
} else {
    <ul>
        <li>@Html.ActionLink("Register", "Register", "Account", routeValues: null,
htmlAttributes: new { id = "registerLink" })</li>
        <li>@Html.ActionLink("Log in", "Login", "Account", routeValues: null, htmlAttributes:
new { id = "loginLink" })</li>
    </ul>
}
```

Student

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Create";
}

<h2>Nuevo Estudiante</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Estudiante</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
```

```
            <div class="editor-field">
                @Html.EditorFor(model => model.CareerCode)
                @Html.ValidationMessageFor(model => model.CareerCode)
            </div>

            <div class="editor-label">
                @Html.Label("FirstName", "Nombres")
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.FirstName)
                @Html.ValidationMessageFor(model => model.FirstName)
            </div>

            <div class="editor-label">
                @Html.Label("LastName", "Apellido")
            </div>
            <div class="editor-field">
                @Html.EditorFor(model => model.LastName)
                @Html.ValidationMessageFor(model => model.LastName)
            </div>
            <p>
                <input type="submit" value="Guardar" />
            </p>
        </fieldset>
    }

    <div>
        @Html.ActionLink("Volver", "Index")
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }
```

Delete.cshtml

```
    @model TesinaMobileCloud.Data.Model.Student

    @{
        ViewBag.Title = "Delete";
    }

    <h2>Eliminar Estudiante</h2>

    <h3>Está seguro que desea eliminar este Estudiante?</h3>
    <fieldset>
        <legend>Estudiante</legend>
```

```
    <div class="display-label">
        @Html.Label("StudentCode", "Código")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("FirstName", "Nombres")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FirstName)
    </div>

    <div class="display-label">
        @Html.Label("LastName", "Apellido")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.LastName)
    </div>
</fieldset>
@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Eliminar" /> |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
```

```
    <legend>Estudiante</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código")
    </div>
    <div class="display-field">
       @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CareerCode", "Código de Carrera")
    </div>
    <div class="display-field">
       @Html.DisplayFor(model => model.CareerCode)
    </div>

    <div class="display-label">
        @Html.Label("FirstName", "Nombres")
    </div>
    <div class="display-field">
       @Html.DisplayFor(model => model.FirstName)
    </div>

    <div class="display-label">
        @Html.Label("LastName", "Apellido")
    </div>
    <div class="display-field">
       @Html.DisplayFor(model => model.LastName)
    </div>
</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new {  id=Model.StudentCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.Student

@{
    ViewBag.Title = "Edit";
}

<h2>Editar</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)
```

```
    <fieldset>
        <legend>Estudiante</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CareerCode", "Código de Carrera")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CareerCode)
            @Html.ValidationMessageFor(model => model.CareerCode)
        </div>

        <div class="editor-label">
            @Html.Label("FirstName", "Nombres")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FirstName)
            @Html.ValidationMessageFor(model => model.FirstName)
        </div>

        <div class="editor-label">
            @Html.Label("LastName", "Apellido")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.LastName)
            @Html.ValidationMessageFor(model => model.LastName)
        </div>
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
```

```
}
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.Student>

@{
    ViewBag.Title = "Index";
}

<h2>Estudiantes</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("CareerCode", "Carrera")
        </th>
        <th>
            @Html.Label("StudentCode", "Matricula")
        </th>
        <th>
            @Html.Label("FirstName", "Nombre")
        </th>
        <th>
            @Html.Label("LastName", "Apellido")
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CareerCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.StudentCode)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.FirstName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.LastName)
        </td>
        <td>
```

```
            @Html.ActionLink("Editar", "Edit", new {  id=item.StudentCode}) |
            @Html.ActionLink("Detalles", "Details", new { id=item.StudentCode }) |
            @Html.ActionLink("Eliminar", "Delete", new { id=item.StudentCode})
        </td>
    </tr>
}

</table>
```

StudentCourseRelationship
AsignAssistedHours.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Asignar Horas Cursadas";
}

<h2>Asignar Horas Cursadas</h2>
<div>
    @using (Html.BeginForm())
    {
        <fieldset>
            <legend>Asignación</legend>
            @Html.HiddenFor(model => model.StudentCode)
            @Html.HiddenFor(model => model.CourseCode)
            <div id="ActualAssitedHours">
                <label>Horas Asistidas a la fecha</label>
                @Html.DisplayFor(model => model.TotalHoursAssisted)
            </div>
            <div id="AddAssitedHours">
                <label>Agregar Horas</label>
                @Html.TextBox("AddHours")
            </div>
            <p>
                <input type="submit" value="Guardar" />
            </p>
        </fieldset>
    }

    <div>
        @Html.ActionLink("Volver", "Index")
    </div>
</div>
```

StudentCourseRelationship

Create.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Create";
}

<h2>Asignación de Materias a Estudiantes</h2>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código de Estudiante")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Materia")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PartialExamResult)
            @Html.ValidationMessageFor(model => model.PartialExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("FinalExamResult", "Calificación de Examen Final")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.FinalExamResult)
            @Html.ValidationMessageFor(model => model.FinalExamResult)
```

```
        </div>

        <div class="editor-label">
            @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RecuperationExamResult)
            @Html.ValidationMessageFor(model => model.RecuperationExamResult)
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PracticWorkResult)
            @Html.ValidationMessageFor(model => model.PracticWorkResult)
        </div>

        <div class="editor-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.TotalHoursAssisted)
            @Html.ValidationMessageFor(model => model.TotalHoursAssisted)
        </div>
        @* <div class="editor-label">
            @Html.Label("FinalExamInscripted", "Inscripto en Examen Final")
        </div>
        <div class="editor-field">
            @Html.CheckBoxFor(model => model.FinalExamInscripted)
        </div>
        <div class="editor-label">
            @Html.Label("IsCurrentCourse", "Cursando")
        </div>
        <div class="editor-field">
            @Html.CheckBoxFor(model => model.IsCurrentCourse)
        </div>*@
        <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>
```

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Delete";
}

<h2>Eliminar Asignación</h2>

<h3>Está seguro que desea eliminar esta asignación?</h3>
<fieldset>
    <legend>Asignación de Estudiante a Materia</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código de Estudiante")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Materia")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
    </div>
    <div class="display-field">
        @if (Model.PartialExamResult != 0)
        {
            @Html.DisplayFor(model => model.PartialExamResult)
        }
        else
        {
            <label>-</label>
        }
    </div>

    <div class="display-label">
```

```
    @Html.Label("FinalExamResult", "Calificación de Examen Final")
</div>
<div class="display-field">
    @if (Model.FinalExamResult != 0)
    {
        @Html.DisplayFor(model => model.FinalExamResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
</div>
<div class="display-field">
    @if (Model.RecuperationExamResult != 0)
    {
        @Html.DisplayFor(model => model.RecuperationExamResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
</div>
<div class="display-field">
    @if (Model.PracticWorkResult != 0)
    {
        @Html.DisplayFor(model => model.PracticWorkResult)
    }
    else
    {
        <label>-</label>
    }
</div>

<div class="display-label">
    @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
</div>
<div class="display-field">
    @Html.DisplayFor(model => model.TotalHoursAssisted)
</div>
```

```
</fieldset>
@using (Html.BeginForm())
{
    <p>
        <input type="submit" value="Eliminar" />
        |
        @Html.ActionLink("Volver", "Index")
    </p>
}
```

Details.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Details";
}

<h2>Detalles</h2>

<fieldset>
    <legend>Asignación Estudiante a Materia</legend>

    <div class="display-label">
        @Html.Label("StudentCode", "Código de Estudiante")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.StudentCode)
    </div>

    <div class="display-label">
        @Html.Label("CourseCode", "Código de Materia")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.CourseCode)
    </div>

    <div class="display-label">
        @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.PartialExamResult)
    </div>

    <div class="display-label">
        @Html.Label("FinalExamResult", "Calificación de Examen Final")
```

```
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.FinalExamResult)
    </div>

    <div class="display-label">
         @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.RecuperationExamResult)
    </div>

    <div class="display-label">
         @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.PracticWorkResult)
    </div>

    <div class="display-label">
         @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
    </div>
    <div class="display-field">
        @Html.DisplayFor(model => model.TotalHoursAssisted)
    </div>

</fieldset>
<p>
    @Html.ActionLink("Editar", "Edit", new {  id=Model.StudentCode }) |
    @Html.ActionLink("Volver", "Index")
</p>
```

Edit.cshtml

```
@model TesinaMobileCloud.Data.Model.StudentCourseRelationship

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
```

```
        <legend>Asignación</legend>

        <div class="editor-label">
            @Html.Label("StudentCode", "Código de Estudiante")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.StudentCode)
            @Html.ValidationMessageFor(model => model.StudentCode)
        </div>

        <div class="editor-label">
            @Html.Label("CourseCode", "Código de Materia")
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.CourseCode)
            @Html.ValidationMessageFor(model => model.CourseCode)
        </div>

        <div class="editor-label">
            @Html.Label("PartialExamResult", "Calificación de Examen Parcial")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.PartialExamResult)
            @if (Model.PartialExamResult != 0)
            {
                @Html.DisplayFor(model => model.PracticWorkResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "PartialExam" })
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("FinalExamResult", "Calificación de Examen Final")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.FinalExamResult)
            @if (Model.FinalExamResult != 0)
            {
                @Html.DisplayFor(model => model.FinalExamResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "FinalExam" })
            }
            else
```

```
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("RecuperationExamResult", "Calificación de Examen Recuperatorio")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.RecuperationExamResult)
            @if (Model.RecuperationExamResult != 0)
            {
                @Html.DisplayFor(model => model.RecuperationExamResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "RecuperationExam" })
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </div>
        <div class="editor-field">
            @Html.HiddenFor(model => model.PracticWorkResult)
            @if (Model.PracticWorkResult != 0)
            {
                @Html.DisplayFor(model => model.PracticWorkResult)
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
Model.StudentCode, course = Model.CourseCode, exam = "PracticWorkResult" })
            }
            else
            {
                <label>-</label>
            }
        </div>

        <div class="editor-label">
            @Html.Label("TotalHoursAssisted", "Cantidad de Horas Asistidas")
        </div>
        <div class="editor-field">
            @Html.DisplayFor(model => model.TotalHoursAssisted)
            @Html.ActionLink("Asignar Horas Cursadas", "AsignAssistedHours", new { id =
Model.StudentCode, course = Model.CourseCode })
        </div>
```

```
            <div class="editor-label">
                @Html.Label("FinalExamInscripted", "Inscripto en Examen Final")
            </div>
            <div class="editor-field">
                @Html.CheckBoxFor(model => model.FinalExamInscripted)
            </div>
            <div class="editor-label">
                @Html.Label("IsCurrentCourse", "Cursando")
            </div>
            <div class="editor-field">
                @Html.CheckBoxFor(model => model.IsCurrentCourse)
            </div>
            <p>
                <input type="submit" value="Guardar" />
            </p>
        </fieldset>
    }

    <div>
        @Html.ActionLink("Volver", "Index")
    </div>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }
```

ExamResultAsign.cshtml

```
    @model TesinaMobileCloud.Data.Model.StudentCourseRelationship

    @{
        ViewBag.Title = "Asignar Calificación de Examen";
    }

    <h2>Asignar Calificación de Examen</h2>
    @using (Html.BeginForm())
    {
        @Html.ValidationSummary(true)

        <fieldset>
            <legend>Asignación</legend>
            @Html.HiddenFor(model => model.StudentCode)
            @Html.HiddenFor(model => model.CourseCode)
            <div class="editor-label">
                @Html.Label("ExamType", "Examen")
            </div>
            <div class="editor-field">
```

```
            @Html.DropDownList("SelectedExamType",
        (List<SelectListItem>)ViewBag.ExamTypes)
        </div>

        <div class="editor-label">
            @Html.Label("ExamResult", "Calificación")
        </div>
        <div class="editor-field">
            @Html.TextBox("ExamResultText")
        </div>
    <p>
            <input type="submit" value="Guardar" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Volver", "Index")
</div>
```

Index.cshtml

```
@model IEnumerable<TesinaMobileCloud.Data.Model.StudentCourseRelationship>

@{
    ViewBag.Title = "Index";
}

<h2>Estado del Estudiante en una Materia</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.Label("StudentCode", "Matricula del Estudiante")
        </th>
        <th>
            @Html.Label("CourseCode", "Código de la Materia")
        </th>
        <th>
            @Html.Label("PartialExamResult", "Resultado de Examen Parcial")
        </th>
        <th>
            @Html.Label("FinalExamResult", "Resultado de Examen Final")
        </th>
        <th>
```

```
            @Html.Label("RecuperationExamResult", "Resultado de Examen Recuperatorio")
        </th>
        <th>
            @Html.Label("PracticWorkResult", "Calificación de Trabajos Practicos")
        </th>
        <th>
            @Html.Label("TotalHoursAssisted", "Total de Horas Asistidas")
        </th>
        <th>
            @Html.Label("FinalExamInscripted", "Inscripto a Examen Final")
        </th>
        <th>
            @Html.Label("IsCurrentCourse", "Cursando esta materia")
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.StudentCode)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.CourseCode)
            </td>
            <td>
                @if (item.PartialExamResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.PartialExamResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @if (item.FinalExamResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.FinalExamResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
```

```
                @if (item.RecuperationExamResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.RecuperationExamResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @if (item.PracticWorkResult != 0)
                {
                    @Html.DisplayFor(modelItem => item.PracticWorkResult)
                }
                else
                {
                    <label>-</label>
                }
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TotalHoursAssisted)
            </td>
            <td>
                @Html.Label("FinalExamInscripted", item.FinalExamInscripted ? "Si" : "No")
            </td>
            <td>
                @Html.Label("IsCurrentCourse", item.IsCurrentCourse ? "Si" : "No")
            </td>
            <td>
                @Html.ActionLink("Asignar Calificación", "ExamResultAsign", new { id =
item.StudentCode, course = item.CourseCode })
                @Html.ActionLink("Asignar Horas Cursadas", "AsignAssistedHours", new { id =
item.StudentCode, course = item.CourseCode })
                @Html.ActionLink("Editar", "Edit", new { id = item.StudentCode, courseCode =
item.CourseCode }) |
                @Html.ActionLink("Detalles", "Details", new { id = item.StudentCode, courseCode
= item.CourseCode }) |
                @Html.ActionLink("Eliminar", "Delete", new { id = item.StudentCode, row =
item.CourseCode })
            </td>
        </tr>
    }

</table>
```

TesinaMobileCloud.Data

CloudStorageConfiguration.cs

```
using System;
using System.Configuration;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace TesinaMobileCloud.Data
{
   public class CloudStorageConfiguration
   {
      public static CloudStorageAccount GetCloudAccount(string
cloudStorageConnectionStringName)
      {
         try
         {
            return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
         }
         catch (InvalidOperationException)
         {
            return SetConfigurationAndGetAccount(cloudStorageConnectionStringName);
         }
      }

      private static CloudStorageAccount SetConfigurationAndGetAccount(string
cloudStorageConnectionStringName)
      {
         SetConfigurationSettingsPublisher();
         return
CloudStorageAccount.FromConfigurationSetting(cloudStorageConnectionStringName);
      }

      private static void SetConfigurationSettingsPublisher()
      {
         CloudStorageAccount.SetConfigurationSettingPublisher((configName,
configSettingPublisher) =>
         {
            var configValue = ConfigurationManager.AppSettings[configName];
            if (RoleEnvironment.IsAvailable)
               configValue = RoleEnvironment.GetConfigurationSettingValue(configName);
            configSettingPublisher(configValue);
         });
      }

   }
}
```

DTO

Attendance.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Attendance
    {
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public DateTime TotalHoursOfClasses { get; set; }
        [DataMember]
        public int ActualPercentage { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Career
    {
        [DataMember]
        public string Code { get; set; }
        [DataMember]
        public string Title { get; set; }
    }
}
```

Course.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Course
    {
        [DataMember]
```

```
        public int Code { get; set; }
        [DataMember]
        public string Title { get; set; }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public int YearOfCareer { get; set; }
        [DataMember]
        public DateTime? PartialExamDate { get; set; }
        [DataMember]
        public DateTime? FinalExamDate { get; set; }
        [DataMember]
        public DateTime? RecuperationExameDate { get; set; }
        [DataMember]
        public DateTime? PracticWorkPresentationDate { get; set; }
        [DataMember]
        public int TotalHoursOfClasses { get; set; }
        [DataMember]
        public int TotalHoursInAWeek { get; set; }
        [DataMember]
        public int TotalHoursDictated { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Scheduled { get; set; }
    }
}
```

Student.cs

```
using System.Collections.Generic;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class Student
    {
        public Student()
        {
            Courses = new List<StudentCourse>();
        }
        [DataMember]
        public IList<StudentCourse> Courses { get; set; }
        [DataMember]
        public Career Career { get; set; }
        [DataMember]
```

```
        public string FirstName { get; set; }
        [DataMember]
        public string LastName { get; set; }
        [DataMember]
        public int StudentCode { get; set; }
        [DataMember]
        public int CareerCode { get; set; }
        [DataMember]
        public string DeviceUri { get; set; }
        [DataMember]
        public string Password { get; set; }
    }
}
```

StudentCourse.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class StudentCourse
    {
        [DataMember]
        public double PartialExamResult { get; set; }
        [DataMember]
        public double FinalExamResult { get; set; }
        [DataMember]
        public double RecuperationExamResult { get; set; }
        [DataMember]
        public double PracticWorkResult { get; set; }
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public Course Course { get; set; }
        [DataMember]
        public bool FinalExamInscripted { get; set; }
        [DataMember]
        public int AttendanceState { get; set; }
        [DataMember]
        public bool IsCurretCourse { get; set; }

        public double Attendance()
        {
            try
            {
```

```
                return (TotalHoursAssisted * 100) / Course.TotalHoursDictated;
            }
            catch (DivideByZeroException)
            {
                return 0;
            }
            //if (result > 75)
            //    return 3;
            //if (result < 75 && result > 50)
            //    return 2;
            //return 3;
        }
    }
}
```

TaskResult.cs

```
using System.Runtime.Serialization;
namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public enum TaskResult
    {
        [EnumMember]
        Success,
        [EnumMember]
        Failed
    }
}
```

TaskSummary.cs

```
using System.Runtime.Serialization;
namespace TesinaMobileCloud.Data.DTO
{
    [DataContract]
    public class TaskSummary
    {
        [DataMember]
        public TaskResult Result { get; set; }
        [DataMember]
        public string Message { get; set; }
    }
}
```

Helpers

AttendanceState.cs

```
namespace TesinaMobileCloud.Data.Helpers
```

```
{
    public enum AttendanceState
    {
        Danger = 1,
        Precaution = 2,
        Good = 3
    }
}
```

JsonSerializationHelper.cs

```
using System.IO;
using System.Runtime.Serialization.Json;
using System.Text;

namespace TesinaMobileCloud.Data.Helpers
{
    public class JsonSerializationHelper
    {
        public static string Serialize<T>(T entity)
        {
            var ser = new DataContractJsonSerializer(typeof(T));
            var ms = new MemoryStream();
            ser.WriteObject(ms, entity);
            var jsonString = Encoding.UTF8.GetString(ms.ToArray());
            ms.Close();
            return jsonString;
        }

        public static T JsonDeserialize<T>(string jsonString)
        {
            var ser = new DataContractJsonSerializer(typeof(T));
            var ms = new MemoryStream(Encoding.UTF8.GetBytes(jsonString));
            T obj = (T)ser.ReadObject(ms);
            return obj;
        }

    }
}
```

Model

AzureEntity.cs

```
using System;
using System.Data.Services.Common;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
```

```
    [DataServiceEntity]
    [DataContract]
    public class AzureEntity
    {
        public string RowKey { get; set; }
        public string PartitionKey { get; set; }
        public DateTime Timestamp { get; set; }
    }
}
```

Career.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Career : AzureEntity
    {
        [DataMember]
        public int CareerCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return RowKey; }
            set { RowKey = value; }
        }

        public Career()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Career(int careerCode, string title)
        {
            CareerCode = careerCode;
            Title = title;
        }


    }
}
```

CareerCourseRelationship.cs

```
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
   [DataContract]
   public class CareerCourseRelationship : AzureEntity
   {
      [DataMember]
      public int CareerCode
      {
         get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
         set { PartitionKey = value.ToString(); }
      }
      [DataMember]
      public int CourseCode
      {
         get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
         set { RowKey = value.ToString(); }
      }
      [DataMember]
      public int YearInTheCareer { get; set; }

      public CareerCourseRelationship()
      {
         PartitionKey = string.Empty;
         RowKey = string.Empty;
      }

      public CareerCourseRelationship(int careerCode, int courseCode)
      {
         CareerCode = careerCode;
         CourseCode = courseCode;
      }
   }
}
```

Course.cs

```
using System;
using System.Runtime.Serialization;

namespace TesinaMobileCloud.Data.Model
{
   [DataContract]
   public class Course : AzureEntity
   {
      [DataMember]
```

```
        public int Code
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public string Title
        {
            get { return string.IsNullOrEmpty(RowKey) ? string.Empty : RowKey; }
            set { RowKey = value; }
        }
        [DataMember]
        public string Professor { get; set; }
        [DataMember]
        public DateTime? FinalExamDate { get; set; }
        [DataMember]
        public DateTime? PartialExameDate { get; set; }
        [DataMember]
        public DateTime? PracticWorkPresentationDate { get; set; }
        [DataMember]
        public DateTime? RecuperationExameDate { get; set; }
        [DataMember]
        public int TotalHoursOfClasses { get; set; }
        [DataMember]
        public int TotalHoursInAWeek { get; set; }
        [DataMember]
        public int TotalHoursDictated { get; set; }
        [DataMember]
        public string Description { get; set; }
        [DataMember]
        public string Scheduled { get; set; }
         [DataMember]
        public bool IsCritical { get; set; }

        public Course()
        {
            PartitionKey = string.Empty;
            RowKey = string.Empty;
        }

        public Course(int courseCode, string name)
        {
            Code = courseCode;
            Title = name;
        }
    }
}
```

Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace TesinaMobileCloud.Data.Model
{
    [DataContract]
    public class Student : AzureEntity
    {
        [DataMember]
        public int StudentCode
        {
            get { return int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CareerCode
        {
            get { return int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public string FirstName { get; set; }
        [DataMember]
        public string LastName { get; set; }
        [DataMember]
        public string DeviceUri { get; set; }

        public Student()
        {
            PartitionKey = "0";
            RowKey = "0";
        }
    }
}
```

StudentCourseRelationship.cs

```
using System;
using System.Runtime.Serialization;
using TesinaMobileCloud.Data.Helpers;

namespace TesinaMobileCloud.Data.Model
```

```
{
    [DataContract]
    public class StudentCourseRelationship : AzureEntity
    {
        [DataMember]
        public int StudentCode
        {
            get { return string.IsNullOrEmpty(PartitionKey) ? 0 : int.Parse(PartitionKey); }
            set { PartitionKey = value.ToString(); }
        }
        [DataMember]
        public int CourseCode
        {
            get { return string.IsNullOrEmpty(RowKey) ? 0 : int.Parse(RowKey); }
            set { RowKey = value.ToString(); }
        }
        [DataMember]
        public double PartialExamResult { get; set; }
        [DataMember]
        public double FinalExamResult { get; set; }
        [DataMember]
        public double RecuperationExamResult { get; set; }
        [DataMember]
        public double PracticWorkResult { get; set; }
        [DataMember]
        public int TotalHoursAssisted { get; set; }
        [DataMember]
        public bool FinalExamInscripted { get; set; }
        [DataMember]
        public bool IsCurrentCourse { get; set; }


        public StudentCourseRelationship()
        {
            PartitionKey = "0";
            RowKey = "0";
        }

        public AttendanceState GetAttendanceState(int totalHoursOfClasses, bool isCritical)
        {
            var attendence = 0;

            if (totalHoursOfClasses > 0)
            {
                attendence = (TotalHoursAssisted * 100) / totalHoursOfClasses;
            }
            if (!isCritical)
```

```
            {
                if (attendence >= 75)
                    return AttendanceState.Good;
                if (attendence < 75 && attendence > 50)
                    return AttendanceState.Precaution;
                return AttendanceState.Danger;
            }
            if (attendence >= 90)
                return AttendanceState.Good;
            if (attendence < 90 && attendence > 75)
                return AttendanceState.Precaution;
            return AttendanceState.Danger;
        }
    }
}
```

Queue

IQueue.cs

```
namespace TesinaMobileCloud.Data.Queue
{
    public interface IQueue<T>
    {
        void AddMessage(T message);
        T GetMessage();
        bool HasMessage { get; }
    }


}
```

NotificationQueue.cs

```
using System;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;
using TesinaMobileCloud.Data.Helpers;

namespace TesinaMobileCloud.Data.Queue
{
    public class NotificationQueue<T> : IQueue<T>
    {
        private readonly CloudQueue _queue;
        private const string _queueName = "notificationqueue";

        public NotificationQueue(CloudStorageAccount cloudStorageAccount)
        {
            var cloudQueueClient = cloudStorageAccount.CreateCloudQueueClient();
```

```
        _queue = cloudQueueClient.GetQueueReference(_queueName);
        _queue.CreateIfNotExist();
    }

    public void AddMessage(T message)
    {
        var messageString = JsonSerializationHelper.Serialize<T>(message);
        _queue.AddMessage(new CloudQueueMessage(messageString));
    }

    public T GetMessage()
    {
        var decodedMessage = _queue.GetMessage();
        _queue.DeleteMessage(decodedMessage);
        return JsonSerializationHelper.JsonDeserialize<T>(decodedMessage.AsString);
    }

    public bool HasMessage {
        get
        {
            _queue.FetchAttributes();
            return _queue.ApproximateMessageCount != 0;
        }
    }
}

public class PushNotificationErrorsQueue<T> : IQueue<T>
{

    private readonly CloudQueue _queue;
    private const string QueueName = "pushnotificationerrorsqueue";

    public PushNotificationErrorsQueue(CloudStorageAccount cloudStorageAccount)
    {
        var cloudQueueClient = cloudStorageAccount.CreateCloudQueueClient();
        _queue = cloudQueueClient.GetQueueReference(QueueName);
        _queue.CreateIfNotExist();
    }


    public void AddMessage(T message)
    {
        _queue.AddMessage(new CloudQueueMessage(message.ToString()));
    }

    public T GetMessage()
    {
```

```
            throw new System.NotImplementedException();
        }

        public bool HasMessage { get; private set; }
    }
}
```

QueueNotificationMessage.cs

```
using System;
using Microsoft.WindowsAzure.StorageClient;
using TesinaMobileCloud.Data.Helpers;

namespace TesinaMobileCloud.Data.Queue
{
    public class QueueNotificationMessage
    {
        public string Destinatary { get; set; }

        public string Type { get; set; }

        public string Payload { get; set; }

        public string Priority
        {
            get { return Type.ToLower().Equals("toast") ? "2" : "1"; }
        }
    }
}
```

Repository

CareerCourseRelationshipRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CareerCourseRelationshipRepository :
ICourseCareerRelationshipRepository
    {
        private readonly ITable<CareerCourseRelationship> _table;

        public CareerCourseRelationshipRepository(ITable<CareerCourseRelationship> table)
        {
            _table = table;
        }
```

```
        public int GetCareerYearOfCourseByCareer(int careerCode, int courseCode)
        {
            return
                _table.GetSingleByCriteria(
                    relationship => relationship.CareerCode == careerCode &&
relationship.CourseCode == courseCode).
                    YearInTheCareer;
        }
    }
}
```

CareerRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CareerRepository : ICareerRepository
    {
        private readonly ITable<Career> _careerTable;

        public CareerRepository(ITable<Career> careerTable)
        {
            _careerTable = careerTable;
        }

        public Career GetCareer(string careerCode)
        {
            return _careerTable.GetSingleByCriteria(career => career.CareerCode ==
int.Parse(careerCode));
        }
    }
}
```

CourseRepository.cs

```
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class CourseRepository : ICourseRepository
    {
```

```
        private readonly ITable<Course> _courseTable;
        private readonly ITable<CareerCourseRelationship> _careerCourseTable;

        public CourseRepository(ITable<Course> courseTable,
    ITable<CareerCourseRelationship> careerCourseTable)
        {
            _courseTable = courseTable;
            _careerCourseTable = careerCourseTable;
        }

        public IList<Course> GetCoursesByCareer(string careerCode)
        {
            var courses = _careerCourseTable.GetByCriteria(c => c.CareerCode ==
    int.Parse(careerCode));
            return courses.Select(careerCourseRelationship =>
    _courseTable.GetSingleByCriteria(c => c.Code ==
    careerCourseRelationship.CourseCode)).ToList();
        }

        public IList<Course> GetAllCourses()
        {
            return _courseTable.GetByCriteria(c => !string.IsNullOrEmpty(c.PartitionKey));
        }

        public void AddCourse(Course course)
        {
            _courseTable.Add(course);
        }

        public Course GetSingleCourse(int courseCode)
        {
            return _courseTable.GetSingleByCriteria(c => c.Code == courseCode);
        }
    }
}
```

StudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class StudentCourseRepository : IStudentCourseRepository
    {
        private readonly ITable<StudentCourseRelationship> _studentCourseTable;
```

```
        public StudentCourseRepository(ITable<StudentCourseRelationship>
    studentCourseTable)
        {
            _studentCourseTable = studentCourseTable;
        }

        public IEnumerable<StudentCourseRelationship> RetriveCoursesOfStudent(string
    studentCode)
        {
            return _studentCourseTable.GetByCriteria(s => s.StudentCode ==
    int.Parse(studentCode));
        }

        public StudentCourseRelationship RetriveStudentCourseByCode(string studentCode,
    string courseCode)
        {
            return _studentCourseTable.GetSingleByCriteria(sc => sc.StudentCode ==
    int.Parse(studentCode) && sc.CourseCode == int.Parse(courseCode));
        }
    }
}
```

StudentRepository.cs

```
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Repository
{
    public class StudentRepository : IStudentRepository
    {
        private readonly ITable<Student> _studentTable;

        public StudentRepository(ITable<Student> studentTable)
        {
            _studentTable = studentTable;
        }

        public Student GetStudentByCareerCodeAndStudentCode(string studentCode, string
    careerCode)
        {
            return _studentTable.GetSingleByCriteria(s => s.StudentCode ==
    int.Parse(studentCode) &&
                                                s.CareerCode == int.Parse(careerCode));
        }
```

```
        public void Add(Student student)
        {
            _studentTable.Add(student);
        }

        public Student GetStudentByStudentCode(string studentCode)
        {
            return _studentTable.GetSingleByCriteria(s => s.StudentCode ==
int.Parse(studentCode));
        }
    }
}
```

Interfaces

ICareerRepository.cs

```
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICareerRepository
    {
        Career GetCareer(string careerCode);
    }
}
```

ICourseCareerRelationshipRepository.cs

```
namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseCareerRelationshipRepository
    {
        int GetCareerYearOfCourseByCareer(int careerCode, int courseCode);
    }
}
```

ICourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface ICourseRepository
    {
        IList<Course> GetCoursesByCareer(string careerCode);
        IList<Course> GetAllCourses();
        void AddCourse(Course course);
```

```
        Course GetSingleCourse(int courseCode);
    }
}
```

IStudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface IStudentCourseRepository
    {
        IEnumerable<StudentCourseRelationship> RetriveCoursesOfStudent(string
studentCode);
        StudentCourseRelationship RetriveStudentCourseByCode(string studentCode, string
courseCode);
    }
}
```

IStudentRepository.cs

```
using TesinaMobileCloud.Data.Model;

namespace TesinaMobileCloud.Data.Repository.Interfaces
{
    public interface IStudentRepository
    {
        Student GetStudentByCareerCodeAndStudentCode(string studentCode, string
careerCode);
        void Add(Student student);
        Student GetStudentByStudentCode(string studentCode);
    }
}
```

Table

AzureTable.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.StorageClient;

namespace TesinaMobileCloud.Data.Table
{
    public class AzureTable<T> : ITable<T>
    {
        private readonly TableServiceContext _context;
```

```
        private readonly string _tableName;
        private IQueryable<T> Query { get; set; }

        public AzureTable(CloudStorageAccount cloudStorageAccount)
        {
            _tableName = typeof (T).Name;

            var table = new CloudTableClient(cloudStorageAccount.TableEndpoint.ToString(),
cloudStorageAccount.Credentials);
            _context = table.GetDataServiceContext();
            table.CreateTableIfNotExist(_tableName);
            Query = _context.CreateQuery<T>(_tableName).AsTableServiceQuery();
        }

        public IList<T> GetByCriteria(Func<T, bool> func)
        {
            return Query.Where(func).ToList();
        }

        public T GetSingleByCriteria(Func<T, bool> func)
        {
            return Query.SingleOrDefault(func);
        }

        public void Add(T entity)
        {
            try
            {
                _context.AddObject(_tableName, entity);
            }
            catch (InvalidOperationException)
            {
                _context.UpdateObject(entity);
            }
            _context.SaveChanges();

        }

        public void Delete(T entity)
        {
            _context.DeleteObject(entity);
            _context.SaveChanges();
        }
    }
}
```

ITable.cs

```
using System;
using System.Collections.Generic;

namespace TesinaMobileCloud.Data.Table
{
   public interface ITable<T>
   {
      IList<T> GetByCriteria(Func<T, bool> func);
      T GetSingleByCriteria(Func<T, bool> func);
      void Add(T entity);
      void Delete(T entity);
   }
}
```

TesinaMobileCloud.Data.Test

CourseCareerRelationshipRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
   [TestClass]
   public class CourseCareerRelationshipRepositoryFixture
   {
      private Mock<ITable<CareerCourseRelationship>> _mockCourseCareerRelationship;

      [TestInitialize]
      public void Setup()
      {
         _mockCourseCareerRelationship = new
Mock<ITable<CareerCourseRelationship>>();
      }

      [TestMethod]
      public void GetCareerYearOfCourseByCareer()
      {
         var career = 502;
         var course = 120;
         _mockCourseCareerRelationship.Setup(
            m => m.GetSingleByCriteria(It.IsAny<Func<CareerCourseRelationship,
bool>>())).Returns(new CareerCourseRelationship
```

```
                                                                {
                                                                    CareerCode = career,
                                                                    CourseCode =
course,
                                                                    YearInTheCareer = 1
                                                                });

            var sut = new
CareerCourseRelationshipRepository(_mockCourseCareerRelationship.Object);
            var result = sut.GetCareerYearOfCourseByCareer(career, course);

            Assert.AreEqual(1, result);


        }
    }
}
```

CourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class CourseRepositoryFixture
    {
        private Mock<ITable<Course>> _mockCourse;
        private Mock<ITable<CareerCourseRelationship>> _mockCareerCourse;

        [TestInitialize]
        public void Setup()
        {
            _mockCourse = new Mock<ITable<Course>>();
            _mockCareerCourse = new Mock<ITable<CareerCourseRelationship>>();
        }

        [TestMethod]
        public void GetCoursesByCareer()
        {
            const int careerCode = 502;
```

```
            _mockCareerCourse.Setup(m =>
m.GetByCriteria(It.IsAny<Func<CareerCourseRelationship, bool>>())).Returns(new
List<CareerCourseRelationship>
                                                                        {
                                                                            new
CareerCourseRelationship(502, 123),
                                                                            new
CareerCourseRelationship(502, 124),
                                                                        });
            _mockCourse.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new Course(123, "Programación II"));
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetCoursesByCareer(careerCode.ToString());

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 2);
        }

        [TestMethod]
        public void GetAllCourses()
        {
            _mockCourse.Setup(m => m.GetByCriteria(It.IsAny<Func<Course,
bool>>())).Returns(new List<Course>{ new Course(123, "Programación II")});
            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);

            var result = sut.GetAllCourses();

            Assert.IsNotNull(result);
            Assert.AreEqual(result.Count, 1);
        }

        [TestMethod]
        public void AddCourse()
        {
            var course = new Course(123, "Programación II");
            _mockCourse.Setup(m => m.Add(It.IsAny<Course>())).Verifiable();

            var sut = new CourseRepository(_mockCourse.Object,
_mockCareerCourse.Object);
            sut.AddCourse(course);
            _mockCourse.Verify();
        }
    }
}
```

StudentCourseRepositoryFixture.cs

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
   [TestClass]
   public class StudentCourseRepositoryFixture
   {
      private Mock<ITable<StudentCourseRelationship>> _studentCourseTable;

      [TestInitialize]
      public void Setup()
      {
         _studentCourseTable = new Mock<ITable<StudentCourseRelationship>>();
      }
      [TestMethod]
      public void RetriveCoursesOfStudent()
      {
         _studentCourseTable.Setup(m =>
m.GetByCriteria(It.IsAny<Func<StudentCourseRelationship, bool>>()))
                        .Returns(new List<StudentCourseRelationship>());
         var sut = new StudentCourseRepository(_studentCourseTable.Object);

         var result = sut.RetriveCoursesOfStudent("10004");

         Assert.IsNotNull(result);
         Assert.IsInstanceOfType(result,
typeof(IEnumerable<StudentCourseRelationship>));
      }

      [TestMethod]
      public void RetriveStudentCourseByCode()
      {
         const string studentCode = "2";
         const string courseCode = "3";
         _studentCourseTable.Setup(m =>
m.GetSingleByCriteria(It.IsAny<Func<StudentCourseRelationship, bool>>()))
                        .Returns(new StudentCourseRelationship
                           {
```

```
                                StudentCode = int.Parse(studentCode),
                                CourseCode = int.Parse(courseCode)
                            });
            var sut = new StudentCourseRepository(_studentCourseTable.Object);

            var result = sut.RetriveStudentCourseByCode(studentCode, courseCode);

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof (StudentCourseRelationship));
            Assert.AreEqual(studentCode, result.StudentCode);
            Assert.AreEqual(courseCode, result.CourseCode);
        }
    }
}
```

StudentRepositoryFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Data.Test
{
    [TestClass]
    public class StudentRepositoryFixture
    {
        private Mock<ITable<Student>> _mockStudentTable;

        [TestInitialize]
        public void Setup()
        {
            _mockStudentTable = new Mock<ITable<Student>>();
        }

        [TestMethod]
        public void GetStudentByCareerCodeAndStudentCode()
        {
            const int studentCode = 10004;
            const int careerCode = 502;
            _mockStudentTable.Setup(m => m.GetSingleByCriteria(It.IsAny<Func<Student,
bool>>()))
                             .Returns(new Student
                                      {
                                          CareerCode = careerCode,
                                          StudentCode = studentCode
```

```
                                    });

                var sut = new StudentRepository(_mockStudentTable.Object);

                var result = sut.GetStudentByCareerCodeAndStudentCode(studentCode.ToString(),
        careerCode.ToString());

                Assert.IsNotNull(result);
                Assert.AreEqual(10004, result.StudentCode);
                Assert.AreEqual(502, result.CareerCode);
            }
        }
    }
```

UnitTest1.cs

```
    using System;
    using Microsoft.VisualStudio.TestTools.UnitTesting;

    namespace TesinaMobileCloud.Data.Test
    {
        [TestClass]
        public class QueueTest
        {
            [TestMethod]
            public void TestMethod1()
            {
            }
        }
    }
```

TesinaMobileCloud.Notifications

IPushNotification.cs

```
    namespace TesinaMobileCloud.Notifications
    {
        public interface IPushNotification
        {
            string Type { get; }
            string Payload { get; }
        }
    }
```

TileNotification.cs

```
    using System;
    namespace TesinaMobileCloud.Notifications
    {
```

```
    public class TileNotification : IPushNotification
    {
        public string BackTitle { get; set; }

        public string BackContent { get; set; }

        private string _backBackgroundImage;

        public string BackBackgroundImage
        {
            get { return _backBackgroundImage; }
            set { _backBackgroundImage = new
Uri(String.Format("/BackgroundImages/{0}.png", value), UriKind.Relative).ToString(); }
        }

        public string Payload
        {
            get
            {
                var tileMessage = string.Format("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                                "<wp:Notification xmlns:wp=\"WPNotification\">" +
                                                    "<wp:Tile >" +
                                                        "<wp:BackgroundImage>" +
                                                        "</wp:BackgroundImage>" +
                                                        "<wp:Count>" +
                                                        "</wp:Count>" +
                                                        "<wp:Title>" +
                                                        "</wp:Title>" +

"<wp:BackBackgroundImage>{0}</wp:BackBackgroundImage>" +
                                                        "<wp:BackTitle>{1}</wp:BackTitle>" +
                                                        "<wp:BackContent>{2}</wp:BackContent>" +
                                                    "</wp:Tile> " +
                                                "</wp:Notification>", BackBackgroundImage, BackTitle,
BackContent);

                return tileMessage;
            }
        }


        public string Type
        {
            get { return "token"; }
        }
    }
}
```

ToastNotification.cs

```
namespace TesinaMobileCloud.Notifications
{
    public class ToastNotification : IPushNotification
    {
        public string Title { get; set; }

        public string Content { get; set; }

        public string Type
        {
            get { return "toast"; }
        }

        public string Payload
        {
            get
            {
                return string.Format("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                                     "<wp:Notification xmlns:wp=\"WPNotification\">" +
                                       "<wp:Toast>" +
                                          "<wp:Text1>{0}</wp:Text1>" +
                                          "<wp:Text2>{1}</wp:Text2>" +
                                          "<wp:Param>{2}</wp:Param>" +
                                       "</wp:Toast>" +
                                     "</wp:Notification>", Title, Content, Page);
            }
        }

        public string Page { get; set; }
    }
}
```

TesinaMobileCloud.Phone.Agents

ScheduledAgent.cs

```
using System;
using System.Windows;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
```

```
using TesinaMobileCloud.Phone.Services.Interfaces;
using Microsoft.Phone.Shell;
using System.Linq;

namespace TesinaMobileCloud.Phone.Agents
{
   public class ScheduledAgent : ScheduledTaskAgent
   {
      private static volatile bool _classInitialized;
      private readonly ICloudService _cloudService;
      private readonly IStudentInformationManager _studentInformationManager;
      private readonly IStudentRepository _studentRepository;

      /// <remarks>
      /// ScheduledAgent constructor, initializes the UnhandledException handler
      /// </remarks>
      public ScheduledAgent()
      {
         if (!_classInitialized)
         {
            _classInitialized = true;
            // Subscribe to the managed exception handler
            Deployment.Current.Dispatcher.BeginInvoke(delegate
            {
               Application.Current.UnhandledException +=
ScheduledAgent_UnhandledException;
            });
         }

         _cloudService = new CloudService();
         _studentRepository = new StudentRepository();
         _studentInformationManager = new
StudentInformationManager(_studentRepository,
                                                new StudentCourseRepository(),
                                                new ScheduleActionServiceAdapter());



      }

      /// Code to execute on Unhandled Exceptions
      private void ScheduledAgent_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
      {
         if (System.Diagnostics.Debugger.IsAttached)
         {
            // An unhandled exception has occurred; break into the debugger
```

```
            System.Diagnostics.Debugger.Break();
        }
    }

    /// <summary>
    /// Agent that runs a scheduled task
    /// </summary>
    /// <param name="task">
    /// The invoked task
    /// </param>
    /// <remarks>
    /// This method is called when a periodic or resource intensive task is invoked
    /// </remarks>
    protected override void OnInvoke(ScheduledTask task)
    {
        if (task is PeriodicTask)
        {
            PerformSyncronizationTask();
        }
    }
    private void PerformSyncronizationTask()
    {
        var syncService = new SyncronizationService(_cloudService,
_studentInformationManager);
        var student = _studentRepository.GetStudent();
        syncService.SyncCourses(student.CareerCode, student.StudentCode,
student.Password)
                .ObserveOnDispatcher()
                .Subscribe(SyncCompleted, SyncFailed);
    }

    private void SyncCompleted(TaskSummary result)
    {
        //DisplayShellNotification(new StandardTileData
        //                {
        //                    Title = "SIA",
        //                    BackContent = "Información Actualizada"
        //                });
        NotifyComplete();
    }

    private void SyncFailed(Exception exception)
    {
        //DisplayShellNotification(new StandardTileData
        //                {
        //                    Title = "SIA",
        //                    BackContent = "Error al sincronizar datos"
```

```
            //                      });
            Abort();
        }

        private static void DisplayShellNotification(ShellTileData tileData)
        {
            var tile = ShellTile.ActiveTiles.First();
            tile.Update(tileData);
        }
    }
}
```

TesinaMobileCloud.Phone.Client

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone" xmlns:sys="clr-
namespace:System;assembly=mscorlib"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.ViewModel"
          xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
          xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client.View"
          mc:Ignorable="d">
    <!--Application Resources-->
    <Application.Resources>
        <ResourceDictionary>
            <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
            <Style TargetType="TextBlock" x:Key="TextBlockHeader">
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="32" />
                <Setter Property="TextWrapping" Value="Wrap"/>
                <Setter Property="FontFamily" Value="Segoe WP Semibold"/>
                <Setter Property="Margin" Value="0,0,0,5"/>
            </Style>
            <Style TargetType="TextBlock" x:Key="TextBlockContent">
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="25" />
                <Setter Property="TextWrapping" Value="Wrap"/>
            </Style>
            <SolidColorBrush x:Key="PhoneControlBackgroundBrush" Color="#FF0083B6"/>
            <Color x:Key="ApplicationBarBrush">#FF0F5CE5</Color>
```

```
            <SolidColorBrush x:Key="ContrastBrush" Color="#FF00518B"/>
            <DataTemplate x:Key="CourseGroupHeaderTemplate">
               <Border Background="Transparent" Padding="0,5,5,5">
                  <Border Background="{StaticResource ContrastBrush}"
BorderBrush="{StaticResource ContrastBrush}" BorderThickness="2" Width="62"
Height="62" Margin="0,0,18,0" HorizontalAlignment="Left">
                     <TextBlock Text="{Binding Key}" Foreground="{StaticResource
PhoneForegroundBrush}" FontSize="48" Padding="6" FontFamily="{StaticResource
PhoneFontFamilySemiLight}" HorizontalAlignment="Left" VerticalAlignment="Center"/>
                  </Border>
               </Border>
            </DataTemplate>
            <DataTemplate x:Key="CourseGroupItemTemplate">
               <Border Background="{StaticResource PhoneControlBackgroundBrush}"
Width="143" Height="113" Margin="6" BorderBrush="{StaticResource
PhoneControlBackgroundBrush}" >
                  <TextBlock Text="{Binding KeyPresentation}" FontFamily="{StaticResource
PhoneFontFamilySemiBold}" FontSize="48" Padding="6"
                                                Foreground="{StaticResource
PhoneForegroundBrush}" VerticalAlignment="Center"/>
               </Border>
            </DataTemplate>
            <DataTemplate x:Key="CourseItemTemplate">
               <local:CourseResumeView Height="85" Width="432"
toolkit:TiltEffect.IsTiltEnabled="True"/>
            </DataTemplate>
         </ResourceDictionary>
      </Application.Resources>
      <Application.ApplicationLifetimeObjects>
         <!--Required object that handles lifetime events for the application-->
         <shell:PhoneApplicationService Launching="Application_Launching"
                                        Closing="Application_Closing"
                                        Activated="Application_Activated"
                                        Deactivated="Application_Deactivated" />
      </Application.ApplicationLifetimeObjects>
   </Application>
```

App.xaml.cs

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client
```

```
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        protected ViewModelLocator Locator
        {
            get { return (ViewModelLocator)Resources["Locator"]; }
        }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            TiltEffect.TiltableItems.Add(typeof(CourseResumeViewModel));
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
                // application's PhoneApplicationService object to Disabled.
```

```
            // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
            // and consume battery power when the user is not using the phone.
            PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
        }

    }

    // Code to execute when the application is launching (eg, from Start)
    // This code will not execute when the application is reactivated
    private void Application_Launching(object sender, LaunchingEventArgs e)
    {
        Locator.ScheduleActions.RemovePeriodicTask();
    }

    // Code to execute when the application is activated (brought to foreground)
    // This code will not execute when the application is first launched
    private void Application_Activated(object sender, ActivatedEventArgs e)
    {

    }

    // Code to execute when the application is deactivated (sent to background)
    // This code will not execute when the application is closing
    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
    {
        // Ensure that required application state is persisted here.
    }

    // Code to execute when the application is closing (eg, user hit Back)
    // This code will not execute when the application is deactivated
    private void Application_Closing(object sender, ClosingEventArgs e)
    {
        Locator.ScheduleActions.AddPeriodicTask();
    }

    // Code to execute if a navigation fails
    private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
    {
        if (System.Diagnostics.Debugger.IsAttached)
        {
            // A navigation has failed; break into the debugger
            System.Diagnostics.Debugger.Break();
        }
    }
```

```
        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new TransitionFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }
```

```
            #endregion
        }
    }
```

MainPage.xaml

```
    <client:PhonePage
        x:Class="TesinaMobileCloud.Phone.Client.MainPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
        xmlns:controls="clr-
    namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:View="clr-namespace:TesinaMobileCloud.Phone.Client.View"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:client="clr-namespace:TesinaMobileCloud.Phone.Client"
        xmlns:toolkit="clr-
    namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
        mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="800"
        FontFamily="{StaticResource PhoneFontFamilyNormal}"
        FontSize="{StaticResource PhoneFontSizeNormal}"
        Foreground="{StaticResource PhoneForegroundBrush}"
        SupportedOrientations="Portrait"
        Orientation="Portrait"
        DataContext="{Binding Main, Source={StaticResource Locator}}"
        shell:SystemTray.IsVisible="False">
        <toolkit:TransitionService.NavigationInTransition>
            <toolkit:NavigationInTransition>
                <toolkit:NavigationInTransition.Backward>
                    <toolkit:TurnstileTransition Mode="BackwardIn"/>
                </toolkit:NavigationInTransition.Backward>
                <toolkit:NavigationInTransition.Forward>
                    <toolkit:TurnstileTransition Mode="ForwardIn"/>
                </toolkit:NavigationInTransition.Forward>
            </toolkit:NavigationInTransition>
        </toolkit:TransitionService.NavigationInTransition>
        <toolkit:TransitionService.NavigationOutTransition>
            <toolkit:NavigationOutTransition>
                <toolkit:NavigationOutTransition.Backward>
                    <toolkit:TurnstileTransition Mode="BackwardOut"/>
                </toolkit:NavigationOutTransition.Backward>
                <toolkit:NavigationOutTransition.Forward>
                    <toolkit:TurnstileTransition Mode="ForwardOut"/>
                </toolkit:NavigationOutTransition.Forward>
            </toolkit:NavigationOutTransition>
        </toolkit:TransitionService.NavigationOutTransition>
```

```
            <client:PhonePage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Mode="Minimized"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="0.35">
            <shell:ApplicationBar.MenuItems>
                    <shell:ApplicationBarMenuItem Text="Configuración" IsEnabled="True"
Click="NavigateToSettingsView"/>
            </shell:ApplicationBar.MenuItems>
            <shell:ApplicationBarIconButton
IconUri="/Assets/AppBar/appbar.feature.search.rest.png" Text="ver todas"
Click="applicationBarIconButton_Click"/>
        </shell:ApplicationBar>
    </client:PhonePage.ApplicationBar>

            <Grid x:Name="LayoutRoot" Background="Transparent">
        <controls:Panorama Title="Sia" Background="{StaticResource
PhoneControlBackgroundBrush}">
            <controls:PanoramaItem Header="Materias">
                <View:CoursesView/>
            </controls:PanoramaItem>
        </controls:Panorama>
    </Grid>
</client:PhonePage>
```

MainPage.xaml.cs

```
using System;
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client
{
    public partial class MainPage : PhonePage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        private void NavigateToSettingsView(object sender, System.EventArgs e)
        {
            var viewModel = DataContext as MainViewModel;

            if (viewModel != null)
                viewModel.NavigateToSettingsView.Execute(null);
        }

        private void applicationBarIconButton_Click(object sender, EventArgs e)
        {
```

```
            var viewModel = DataContext as MainViewModel;

            if (viewModel != null)
                viewModel.NavigateToAllCoursesView.Execute(null);
        }
    }
}
```

PhonePage.cs

```
using System.Windows.Navigation;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client
{
    public class PhonePage : PhoneApplicationPage
    {
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            var dataContext = DataContext as ViewModel.ViewModel;
            if (e.NavigationMode == NavigationMode.Back) return;

            if (dataContext != null)
                dataContext.OnNavigateTo(NavigationContext.QueryString);

            base.OnNavigatedTo(e);
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            var dataContext = DataContext as ViewModel.ViewModel;
            if (e.NavigationMode == NavigationMode.Forward) return;

            if (dataContext != null)
                dataContext.OnNavigateFrom(NavigationContext.QueryString);

            base.OnNavigatedFrom(e);
        }
    }
}
```

Model

GroupCourses.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using TesinaMobileCloud.Phone.Client.ViewModel;
```

```
namespace TesinaMobileCloud.Phone.Client.Model
{
  public class GroupCourses<T> : List<T> where T : CourseResumeViewModel
   {
       public string Key { get; private set; }

      public string KeyPresentation
      {
          get { return string.Format("{0} Año", Key); }
      }

       public GroupCourses(string key)
       {
           Key = key;
       }

       private static List<GroupCourses<T>> CreateGroups()
       {
           return GroupDisplayNames.Select(key => new GroupCourses<T>(key)).ToList();
       }

      protected static IEnumerable<string> GroupDisplayNames
      {
          get { return new[] {"1", "2", "3", "4", "5"}; }
      }

       public static List<GroupCourses<T>> CreateGroups(IEnumerable<T> items)
       {
           var list = CreateGroups();

           foreach (var group in list)
           {
               items.Where(i => i.YearInCareer.Equals(group.Key)).ToList().ForEach(item =>
group.Add(item));
           }
           return list;
       }

   }
}
```

View

AllCoursesView.xaml

```
<Client:PhonePage
```

```
    x:Class="TesinaMobileCloud.Phone.Client.View.AllCoursesView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:Client="clr-namespace:TesinaMobileCloud.Phone.Client"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"  Orientation="Portrait"
    DataContext="{Binding AllCourses, Source={StaticResource Locator}}"
    Opacity="1"
    Background="{StaticResource PhoneControlBackgroundBrush}"
    BorderBrush="{StaticResource PhoneControlBackgroundBrush}"
    shell:SystemTray.IsVisible="True"
          shell:SystemTray.BackgroundColor="{StaticResource ApplicationBarBrush}"
    shell:SystemTray.Opacity="0">

    <toolkit:TransitionService.NavigationInTransition>
        <toolkit:NavigationInTransition>
            <toolkit:NavigationInTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardIn"/>
            </toolkit:NavigationInTransition.Backward>
            <toolkit:NavigationInTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardIn"/>
            </toolkit:NavigationInTransition.Forward>
        </toolkit:NavigationInTransition>
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
        <toolkit:NavigationOutTransition>
            <toolkit:NavigationOutTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardOut"/>
            </toolkit:NavigationOutTransition.Backward>
            <toolkit:NavigationOutTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardOut"/>
            </toolkit:NavigationOutTransition.Forward>
        </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

    <!--LayoutRoot is the root grid where all page content is placed-->
```

```
    <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
        <!--Pivot Control-->
        <controls:Pivot Title="MATERIAS" FontSize="26">
            <controls:PivotItem Header="En Curso">
                <Grid x:Name="CurrentCourses">
                    <toolkit:LongListSelector x:Name="CoursesSelector"
                                ItemsSource="{Binding CurrentCourses}"
                                Background="{x:Null}"
                                GroupHeaderTemplate="{StaticResource
CourseGroupHeaderTemplate}"
                                GroupItemTemplate ="{StaticResource CourseGroupItemTemplate}"
                                ItemTemplate ="{StaticResource CourseItemTemplate}" Margin="0">
                    </toolkit:LongListSelector>
                </Grid>
            </controls:PivotItem>
            <controls:PivotItem Header="Cursadas">
                <Grid x:Name="PastCourses">
                    <toolkit:LongListSelector x:Name="PastCoursesSelector"
                                ItemsSource="{Binding PastCourses}"
                                Background="{x:Null}"
                                GroupHeaderTemplate="{StaticResource
CourseGroupHeaderTemplate}"
                                GroupItemTemplate ="{StaticResource CourseGroupItemTemplate}"
                                ItemTemplate ="{StaticResource CourseItemTemplate}">
                    </toolkit:LongListSelector>
                </Grid>
            </controls:PivotItem>
            <controls:PivotItem Header="Plan de Estudio">
                <Grid x:Name="AllCourses">
                    <toolkit:LongListSelector x:Name="AllCoursesSelector"
                                ItemsSource="{Binding AllCourses}"
                                Background="{x:Null}"
                                GroupHeaderTemplate="{StaticResource
CourseGroupHeaderTemplate}"
                                GroupItemTemplate ="{StaticResource CourseGroupItemTemplate}"
                                ItemTemplate ="{StaticResource CourseItemTemplate}">
                    </toolkit:LongListSelector>
                </Grid>
            </controls:PivotItem>
        </controls:Pivot>
    </Grid>

</Client:PhonePage>
```

AllCoursesView.xaml.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class AllCoursesView : PhonePage
    {
        public AllCoursesView()
        {
            InitializeComponent();
        }
    }
}
```

CourseDescriptionView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client"
x:Class="TesinaMobileCloud.Phone.Client.CourseDescriptionView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    DataContext="{Binding CourseDescription, Source={StaticResource Locator}}"
    d:DesignHeight="607" d:DesignWidth="456">

    <Grid x:Name="LayoutRoot">
        <ListBox HorizontalAlignment="Left" Height="607" VerticalAlignment="Top"
Width="456">
            <TextBlock x:Name="Description" Margin="0,0,0,10" Style="{StaticResource
TextBlockContent}" Text="{Binding Description}" Width="456"/>
            <TextBlock x:Name="Professor" Style="{StaticResource TextBlockHeader}"
Text="Docente"/>
            <TextBlock x:Name="ProfessorName" Text="{Binding Professor}"
Style="{StaticResource TextBlockContent}"/>
```

```
            <TextBlock x:Name="DayAndTime" Style="{StaticResource TextBlockHeader}"
        Text="Horario"/>
            <TextBlock x:Name="DayAndTimeContent" Text="{Binding Scheduled}"
        Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PartialExamDate" Style="{StaticResource TextBlockHeader}"
        Text="Examen Parcial"/>
            <TextBlock x:Name="PartialExamDateContent" Text="{Binding PartialExamDate}"
        Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="RecuperationExameDate" Text="Examen Recuperatorio"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="RecuperationExameDateContent" Text="{Binding
        RecuperationExamDate}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="FinalExamDate" Text="Examen Final" Style="{StaticResource
        TextBlockHeader}"/>
            <TextBlock x:Name="FinalExamDateContent" Text="{Binding FinalExamDate}"
        Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="PracticWorkExameDate" Text="Trabajos Practicos"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="PracticWorkExameDateContent" Text="{Binding
        PracticWorkExamDate}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="TotalHoursOfWeek" Text="Carga Horaria Semanal"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="TotalHoursOfWeekContent" Text="{Binding
        TotalHoursInAWeek}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="TotalHoursOfClasses" Text="Carga Horaria Total"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="TotalHoursOfClassesContent" Text="{Binding
        TotalHoursOfClasses}" Style="{StaticResource TextBlockContent}"/>
            <TextBlock x:Name="TotalHoursDictated" Text="Carga Horaria Dictada"
        Style="{StaticResource TextBlockHeader}"/>
            <TextBlock x:Name="TotalHoursDictatedContent" Text="{Binding
        TotalHoursDictated}" Style="{StaticResource TextBlockContent}"/>

        </ListBox>
    </Grid>
</UserControl>
```

CourseDescriptionView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
```

```
namespace TesinaMobileCloud.Phone.Client
{
   public partial class CourseDescriptionView : UserControl
   {
      public CourseDescriptionView()
      {
         InitializeComponent();
      }
   }
}
```

CourseResumeView.xaml

```
<UserControl
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:i="clr-
namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
xmlns:ec="clr-
namespace:Microsoft.Expression.Interactivity.Core;assembly=Microsoft.Expression.Interact
ions" xmlns:Command="clr-
namespace:GalaSoft.MvvmLight.Command;assembly=GalaSoft.MvvmLight.Extras.WP71"
x:Class="TesinaMobileCloud.Phone.Client.View.CourseResumeView"
        mc:Ignorable="d"
        FontFamily="{StaticResource PhoneFontFamilyNormal}"
        FontSize="{StaticResource PhoneFontSizeNormal}"
        Foreground="{StaticResource PhoneForegroundBrush}"
        d:DesignHeight="85" d:DesignWidth="432" RenderTransformOrigin="0.5,0.5">

        <Grid x:Name="LayoutRoot" Margin="0,0,0,10">
                <Grid Height="75" VerticalAlignment="Top">
        <i:Interaction.Triggers>
           <i:EventTrigger EventName="Tap">
              <Command:EventToCommand Command="{Binding CoursePageDetails}"/>
           </i:EventTrigger>
        </i:Interaction.Triggers>
        <TextBlock x:Name="Title" HorizontalAlignment="Left" Margin="25,-9,0,0"
VerticalAlignment="Top" Height="52" FontSize="37" Text="{Binding Title}"/>
                        <TextBlock x:Name="Professor" HorizontalAlignment="Left"
Margin="25,48,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Text="{Binding
Professor}"/>
                        <Border x:Name="Attendance" Background="{Binding
AttendanceState}" HorizontalAlignment="Left" Height="75" VerticalAlignment="Top"
Width="20"/>
```

```
                    </Grid>
                </Grid>
            </UserControl>
```

CourseResumeView.xaml.cs

```
            using System;
            using System.Collections.Generic;
            using System.Linq;
            using System.Net;
            using System.Windows;
            using System.Windows.Controls;
            using System.Windows.Navigation;
            using Microsoft.Phone.Controls;
            using Microsoft.Phone.Shell;

            namespace TesinaMobileCloud.Phone.Client.View
            {
                public partial class CourseResumeView : UserControl
                {
                    public CourseResumeView()
                    {
                        InitializeComponent();
                    }
                }
            }
```

CourseStudentSituationView.xaml

```
            <UserControl x:Class="TesinaMobileCloud.Phone.Client.CourseStudentSituationView"
                xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
                mc:Ignorable="d"
                FontFamily="{StaticResource PhoneFontFamilyNormal}"
                FontSize="{StaticResource PhoneFontSizeNormal}"
                Foreground="{StaticResource PhoneForegroundBrush}"
                d:DesignHeight="607" d:DesignWidth="456">

                <Grid x:Name="LayoutRoot">
                    <ListBox HorizontalAlignment="Left" Height="607" VerticalAlignment="Top"
            Width="456">
                    <TextBlock x:Name="PartialExamResult" Text="Resultado Examen Parcial"
            Style="{StaticResource TextBlockHeader}" Margin="0"/>
                    <TextBlock x:Name="PartialExamResultContent" Text="{Binding
            PartialExamResult}" Style="{StaticResource TextBlockContent}"/>
```

```
        <TextBlock x:Name="RecuperationExameResult" Style="{StaticResource
TextBlockHeader}">
                        <Run Text="Calificación "/>
                        <Run Text="Examen Recup"/>
                        <Run Text="."/>
                </TextBlock>
        <TextBlock x:Name="RecuperationExameResultContent" Text="{Binding
RecuperationExamResult}" Style="{StaticResource TextBlockContent}"/>
        <TextBlock x:Name="FinalExamResult" Text="Resultado Examen Final"
Style="{StaticResource TextBlockHeader}"/>
        <TextBlock x:Name="FinalExamResultContent" Text="{Binding FinalExamResult}"
Style="{StaticResource TextBlockContent}"/>
        <TextBlock x:Name="PracticWorkExameResult" Style="{StaticResource
TextBlockHeader}">
                        <Run Text="Resultado"/>
                        <Run Text=" "/>
                        <Run Text="de Trab"/>
                        <Run Text="."/>
                        <Run Text=" Practicos"/>
                </TextBlock>
        <TextBlock x:Name="PracticWorkExameResultContent" Text="{Binding
PracticWorkExamResult}" Style="{StaticResource TextBlockContent}"/>
        <TextBlock x:Name="AssistedHoursOfClasses" Text="Total de Horas Asistidas"
Style="{StaticResource TextBlockHeader}"/>
        <TextBlock x:Name="AssistedHoursOfClassesContent" Text="{Binding
TotalAssistedHours}" Style="{StaticResource TextBlockContent}"/>
        <TextBlock x:Name="AssistedPercent" Text="Porcentaje Acumulado de
Asistencias" Style="{StaticResource TextBlockHeader}"/>
        <TextBlock x:Name="AssistedPercentContent" Text="{Binding
ActualAssistedPercent}" Style="{StaticResource TextBlockContent}"/>
         </ListBox>

    </Grid>
</UserControl>
```

CourseStudentSituationView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client
```

```
{
    public partial class CourseStudentSituationView : UserControl
    {
        public CourseStudentSituationView()
        {
            InitializeComponent();
        }
    }
}
```

CoursesView.xaml

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:local="clr-namespace:TesinaMobileCloud.Phone.Client.View"
    x:Class="TesinaMobileCloud.Phone.Client.View.CoursesView"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="498" d:DesignWidth="420"
    DataContext="{Binding Courses, Source={StaticResource Locator}}">
    <Grid x:Name="Courses">
        <toolkit:LongListSelector x:Name="Selector"
                                  ItemsSource="{Binding Courses}"
                                  Background="{x:Null}"
                                  GroupHeaderTemplate="{StaticResource
CourseGroupHeaderTemplate}"
                                  GroupItemTemplate ="{StaticResource CourseGroupItemTemplate}"
                                  ItemTemplate ="{StaticResource CourseItemTemplate}">
        </toolkit:LongListSelector>
    </Grid>
</UserControl>
```

CoursesView.xaml.cs

```
using System.Windows.Controls;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CoursesView : UserControl
    {
        public CoursesView()
```

```
        {
            InitializeComponent();
        }
    }
}
```

CourseView.xaml

```
<Client:PhonePage xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                  xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
                  xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
                  xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
                  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                  xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                  xmlns:Client="clr-namespace:TesinaMobileCloud.Phone.Client"
                  xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    xmlns:ec="clr-
namespace:Microsoft.Expression.Interactivity.Core;assembly=Microsoft.Expression.Interact
ions"
    x:Class="TesinaMobileCloud.Phone.Client.View.CourseView"
    d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    DataContext="{Binding Course, Source={StaticResource Locator}}"
    SupportedOrientations="Portrait"
    Orientation="Portrait"
    mc:Ignorable="d"
    Opacity="1"
    Foreground="{StaticResource PhoneControlBackgroundBrush}"
    Background="{StaticResource PhoneControlBackgroundBrush}"
    BorderBrush="{StaticResource PhoneControlBackgroundBrush}"
    shell:SystemTray.IsVisible="True"
          shell:SystemTray.BackgroundColor="{StaticResource ApplicationBarBrush}"
    shell:SystemTray.Opacity="0">
    <toolkit:TransitionService.NavigationInTransition>
        <toolkit:NavigationInTransition>
            <toolkit:NavigationInTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardIn"/>
            </toolkit:NavigationInTransition.Backward>
            <toolkit:NavigationInTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardIn"/>
            </toolkit:NavigationInTransition.Forward>
```

```
        </toolkit:NavigationInTransition>
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
        <toolkit:NavigationOutTransition>
            <toolkit:NavigationOutTransition.Backward>
                <toolkit:TurnstileTransition Mode="BackwardOut"/>
            </toolkit:NavigationOutTransition.Backward>
            <toolkit:NavigationOutTransition.Forward>
                <toolkit:TurnstileTransition Mode="ForwardOut"/>
            </toolkit:NavigationOutTransition.Forward>
        </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

    <Client:PhonePage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="1">
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Inscribirse a examen final"
Click="CheckForFinalExam" IsEnabled="True"/>
            </shell:ApplicationBar.MenuItems>

                                <shell:ApplicationBarIconButton
IconUri="/Assets/AppBar/appbar.refresh.rest.png" Text="Actualizar"
Click="ApplicationBarIconButton_Click_1"/>
        </shell:ApplicationBar>
    </Client:PhonePage.ApplicationBar>
    <!--LayoutRoot is the root grid where all page content is placed-->
    <shell:SystemTray.ProgressIndicator>
        <shell:ProgressIndicator Text="Actualizando" IsVisible="{Binding Updating}"
IsIndeterminate="True" />
    </shell:SystemTray.ProgressIndicator>
    <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
        <!--Pivot Control-->
        <!--<ProgressBar Background="{StaticResource PhoneControlBackgroundBrush}"
Foreground="#FF033A9B" VerticalAlignment="Top" IsIndeterminate="True"/>-->
        <!--<ProgressBar x:Name="ProgressBar" Background="{StaticResource
PhoneControlBackgroundBrush}" Foreground="#FF033A9B" IsIndeterminate="True"
VerticalAlignment="Top"/>-->
        <controls:Pivot Title="{Binding Title}" SelectedIndex="1" Margin="0,15,0,0"
FontSize="26">
            <controls:PivotItem Header="Descripción">
                <Client:CourseDescriptionView DataContext="{Binding
CourseDescriptionViewModel}"/>
            </controls:PivotItem>
            <!--Pivot item two-->
            <controls:PivotItem Header="Mi Situación">
```

```
                <Client:CourseStudentSituationView DataContext="{Binding
CourseStudentSituationViewModel}"/>
            </controls:PivotItem>
        </controls:Pivot>
    </Grid>
</Client:PhonePage>
```

CourseView.xaml.cs

```
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class CourseView : PhonePage
    {
        public CourseView()
        {
            InitializeComponent();
        }

        private void ApplicationBarIconButton_Click_1(object sender, System.EventArgs e)
        {
            var dataContext = DataContext as CourseViewModel;
            if (dataContext != null)
            {
                dataContext.SyncCourseInfo.Execute(null);
            }
        }

        private void CheckForFinalExam(object sender, System.EventArgs e)
        {
             var dataContext = DataContext as CourseViewModel;
            if (dataContext != null)
            {
                dataContext.CheckForFinalExam.Execute(null);
            }
        }
    }
}
```

SettingsView.xaml

```
<client:PhonePage xmlns:Primitives="clr-
namespace:Microsoft.Phone.Controls.Primitives;assembly=Microsoft.Phone.Controls.Toolki
t"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
```

```
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
                xmlns:client="clr-namespace:TesinaMobileCloud.Phone.Client"
                x:Class="TesinaMobileCloud.Phone.Client.View.SettingsView"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d"
    DataContext="{Binding Settings, Source={StaticResource Locator}}"
    shell:SystemTray.IsVisible="True"
          shell:SystemTray.BackgroundColor="{StaticResource ApplicationBarBrush}"
    shell:SystemTray.Opacity="0">
          <toolkit:TransitionService.NavigationInTransition>
       <toolkit:NavigationInTransition>
          <toolkit:NavigationInTransition.Backward>
             <toolkit:TurnstileTransition Mode="BackwardIn"/>
          </toolkit:NavigationInTransition.Backward>
          <toolkit:NavigationInTransition.Forward>
             <toolkit:TurnstileTransition Mode="ForwardIn"/>
          </toolkit:NavigationInTransition.Forward>
       </toolkit:NavigationInTransition>
    </toolkit:TransitionService.NavigationInTransition>
    <toolkit:TransitionService.NavigationOutTransition>
       <toolkit:NavigationOutTransition>
          <toolkit:NavigationOutTransition.Backward>
             <toolkit:TurnstileTransition Mode="BackwardOut"/>
          </toolkit:NavigationOutTransition.Backward>
          <toolkit:NavigationOutTransition.Forward>
             <toolkit:TurnstileTransition Mode="ForwardOut"/>
          </toolkit:NavigationOutTransition.Forward>
       </toolkit:NavigationOutTransition>
    </toolkit:TransitionService.NavigationOutTransition>

    <client:PhonePage.ApplicationBar>
       <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True"
BackgroundColor="{StaticResource ApplicationBarBrush}" Opacity="1">
          <shell:ApplicationBarIconButton
IconUri="/Toolkit.Content/ApplicationBar.Check.png" Text="Guardar"
Click="SaveChangesInSettings"/>
       </shell:ApplicationBar>
    </client:PhonePage.ApplicationBar>
    <shell:SystemTray.ProgressIndicator>
       <shell:ProgressIndicator Text="Actualizando" IsVisible="{Binding Updating}"
IsIndeterminate="True" />
```

```
    </shell:SystemTray.ProgressIndicator>
    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="Configuración" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}" FontSize="65"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <toolkit:ToggleSwitch HorizontalAlignment="Left" VerticalAlignment="Top"
Margin="0,58,0,0" Width="444" IsChecked="{Binding IsChecked, Mode=TwoWay}"/>
            <TextBlock HorizontalAlignment="Left" Margin="0,10,0,0" TextWrapping="Wrap"
Text="Recibir Notificaciones" VerticalAlignment="Top" FontSize="32"/>
        </Grid>
    </Grid>

</client:PhonePage>
```

SettingsView.xaml.cs

```
using Microsoft.Phone.Controls;
using TesinaMobileCloud.Phone.Client.ViewModel;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class SettingsView : PhonePage
    {
        public SettingsView()
        {
            InitializeComponent();
        }

        private void SaveChangesInSettings(object sender, System.EventArgs e)
        {
            var dataContext = DataContext as SettingsViewModel;
            if (dataContext != null)
                dataContext.SaveChangesInSettings.Execute(null);
        }
    }
}
```

UserCredentialsView.xaml

```
<phone:PhoneApplicationPage
    x:Class="TesinaMobileCloud.Phone.Client.View.UserCredentialsView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="{StaticResource
PhoneControlBackgroundBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="SIA Mobile" Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock Text="Login" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock HorizontalAlignment="Left" Height="34" TextWrapping="Wrap"
Text="Matricula" VerticalAlignment="Top" Width="446" FontSize="26.667"/>
            <TextBox HorizontalAlignment="Left" Height="58" Margin="0,39,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="446" FontSize="18.667"/>
            <TextBlock HorizontalAlignment="Left" Height="34" TextWrapping="Wrap"
VerticalAlignment="Top" Width="446" FontSize="26.667" Margin="0,99,0,0">
                    <Run Text="Carrera"/>
                    <LineBreak/>
                    <Run/>
            </TextBlock>
            <TextBox HorizontalAlignment="Left" Height="58" Margin="0,138,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="446" FontSize="18.667"/>
            <TextBlock HorizontalAlignment="Left" Height="34" TextWrapping="Wrap"
VerticalAlignment="Top" Width="446" FontSize="26.667" Margin="0,202,0,0">
```

```
                    <Run Text="Password"/>
                    <LineBreak/>
                    <Run/>
                </TextBlock>
                <PasswordBox HorizontalAlignment="Left" Height="58" Margin="0,248,0,0"
VerticalAlignment="Top" Width="446"/>
                <Button Content="Ingresar" HorizontalAlignment="Left" Height="82"
Margin="103,364,0,0" VerticalAlignment="Top" Width="248"/>
            </Grid>
        </Grid>

    </phone:PhoneApplicationPage>
```

UserCredentialsView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.View
{
    public partial class UserCredentialsView : PhoneApplicationPage
    {
        public UserCredentialsView()
        {
            InitializeComponent();
        }
    }
}
```

ViewModel

AllCoursesViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Windows;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Client.Model;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
```

```
namespace TesinaMobileCloud.Phone.Client.ViewModel
{
   public class AllCoursesViewModel : ViewModel
   {
      private IStudentRepository _studentRepository;

      public AllCoursesViewModel(IStudentRepository studentRepository)
      {
         _studentRepository = studentRepository;
         CurrentCourses = new
ObservableCollection<GroupCourses<CourseResumeViewModel>>();
         PastCourses = new
ObservableCollection<GroupCourses<CourseResumeViewModel>>();
         AllCourses = new
ObservableCollection<GroupCourses<CourseResumeViewModel>>();
      }

      private void BuildResumeCourses(Student student)
      {
         if (student == null) return;

         var courses = student.Courses;
         if (courses == null) return;

         Deployment.Current.Dispatcher.BeginInvoke(() =>
            {
               var studentCoursesInCurrentCourse = courses.Where(c =>
c.IsCurretCourse).ToList();
               BuildResumeCourse(studentCoursesInCurrentCourse, CurrentCourses,
student);
            });
         Deployment.Current.Dispatcher.BeginInvoke(() =>
            {
               var studentCoursesCoursed =
                  courses.Where(
                     c =>
                     !c.IsCurretCourse &&
                     ((c.PartialExamResult != 0 || c.RecuperationExamResult != 0) &&
c.Attendance() > 0))
                        .ToList();
               BuildResumeCourse(studentCoursesCoursed, PastCourses, student);
            });
         Deployment.Current.Dispatcher.BeginInvoke(() =>
            {
               var studentCoursesCoursed = courses.ToList();
               BuildResumeCourse(studentCoursesCoursed, AllCourses, student);
            });
```

```
        }

        private static void BuildResumeCourse(List<StudentCourse>
    studentCoursesInCurrentCourse,
    ObservableCollection<GroupCourses<CourseResumeViewModel>> currentCourses,
    Student student)
        {
            var courseResumeList = new List<CourseResumeViewModel>();
            studentCoursesInCurrentCourse
                .ForEach(c => courseResumeList.Add(new CourseResumeViewModel
                {
                    Title = c.Course.Title,
                    Code = c.Course.Code,
                    Professor = c.Course.Professor,
                    Attendance = c.Attendance(),
                    AttendanceLevel = c.AttendanceState,
                    StudentCode = student.StudentCode,
                    CareerCode = student.CareerCode,
                    YearInCareer = c.Course.YearOfCareer.ToString()
                }));

            GroupCourses<CourseResumeViewModel>.CreateGroups(courseResumeList)
                                        .ToList()
                                        .ForEach(g => currentCourses.Add(g));
        }

        public ObservableCollection<GroupCourses<CourseResumeViewModel>>
    PastCourses { get; set; }
        public ObservableCollection<GroupCourses<CourseResumeViewModel>>
    CurrentCourses { get; set; }
        public ObservableCollection<GroupCourses<CourseResumeViewModel>> AllCourses
    { get; set; }

        public override void OnNavigateTo(IDictionary<string, string> queryString)
        {
            PastCourses.Clear();
            CurrentCourses.Clear();
            AllCourses.Clear();
            BuildResumeCourses(_studentRepository.GetStudent());
        }
    }
}
```

CourseDescriptionViewModel.cs

```
using System;
using GalaSoft.MvvmLight;
```

```
namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseDescriptionViewModel : ViewModel
    {
        private string _textToDisplayIfDateTimeIsNotValid = "A Confirmar...";

        /// <summary>
        /// The <see cref="Description" /> property's name.
        /// </summary>
        public const string DescriptionPropertyName = "Description";

        private string _description = string.Empty;

        /// <summary>
        /// Sets and gets the Description property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Description
        {
            get
            {
                return _description;
            }
            set
            {
                Set(() => Description, ref _description, value);
            }
        }


        /// <summary>
        /// The <see cref="Professor" /> property's name.
        /// </summary>
        public const string ProfessorPropertyName = "Professor";

        private string _professor = string.Empty;

        /// <summary>
        /// Sets and gets the Professor property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string Professor
        {
            get
            {
                return _professor;
```

```
    }
    set
    {
        Set(() => Professor, ref _professor, value);
    }
}

/// <summary>
/// The <see cref="Scheduled" /> property's name.
/// </summary>
public const string ScheduledPropertyName = "Scheduled";

private string _scheduled = string.Empty;

/// <summary>
/// Sets and gets the Scheduled property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Scheduled
{
    get
    {
        return _scheduled;
    }

    set
    {
        if (_scheduled == value)
        {
            return;
        }

        RaisePropertyChanging(() => Scheduled);
        _scheduled = value;
        RaisePropertyChanged(() => Scheduled);
    }
}

/// <summary>
/// The <see cref="PartialExamDate" /> property's name.
/// </summary>
public const string PartialExamDatePropertyName = "PartialExamDate";

private string _partial = string.Empty;

/// <summary>
/// Sets and gets the PartialExamDate property.
```

```
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string PartialExamDate
        {
            get
            {
                return _partial;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => PartialExamDate, ref _partial, _textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PartialExamDate, ref _partial, value);
            }
        }

        /// <summary>
        /// The <see cref="RecuperationExamDate" /> property's name.
        /// </summary>
        public const string RecuperationExamDatePropertyName = "RecuperationExamDate";

        private string _recuperationExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the RecuperationExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string RecuperationExamDate
        {
            get
            {
                return _recuperationExamDate;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => RecuperationExamDate, ref _recuperationExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => RecuperationExamDate, ref _recuperationExamDate, value);
            }
        }
```

```
        /// <summary>
        /// The <see cref="FinalExamDate" /> property's name.
        /// </summary>
        public const string FinalExamDatePropertyName = "FinalExamDate";

        private string _finalExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the FinalExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string FinalExamDate
        {
            get
            {
                return _finalExamDate;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => FinalExamDate, ref _finalExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => FinalExamDate, ref _finalExamDate, value);
            }
        }

        /// <summary>
        /// The <see cref="PracticWorkExamDate" /> property's name.
        /// </summary>
        public const string PracticWorkExamDatePropertyName = "PracticWorkExamDate";

        private string _practicWorkExamDate = string.Empty;

        /// <summary>
        /// Sets and gets the PracticWorkExamDate property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string PracticWorkExamDate
        {
            get
            {
                return _practicWorkExamDate;
            }
```

```
            set
            {
                if (string.IsNullOrEmpty(value))
                {
                    Set(() => PracticWorkExamDate, ref _practicWorkExamDate,
_textToDisplayIfDateTimeIsNotValid);
                }
                else
                    Set(() => PracticWorkExamDate, ref _practicWorkExamDate, value);
            }
        }

        /// <summary>
        /// The <see cref="TotalHoursOfClasses" /> property's name.
        /// </summary>
        public const string TotalHoursOfClassesPropertyName = "TotalHoursOfClasses";

        private string _totalHoursOfClasses;

        /// <summary>
        /// Sets and gets the TotalHoursOfClasses property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string TotalHoursOfClasses
        {
            get
            {
                return _totalHoursOfClasses;
            }
            set
            {
                Set(() => TotalHoursOfClasses, ref _totalHoursOfClasses, value);
            }
        }

        public const string TotalHoursInAWeekPropertyName = "TotalHoursInAWeek";

        private string _totalHoursInAWeek;

        /// <summary>
        /// Sets and gets the TotalHoursOfClasses property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string TotalHoursInAWeek
        {
            get
            {
```

```
                return _totalHoursInAWeek;
            }
            set
            {
                Set(() => TotalHoursInAWeek, ref _totalHoursInAWeek, value);
            }
        }

        public const string TotalHoursDictatedPropertyName = "TotalHoursDictated";

        private string _totalHoursDictated;

        /// <summary>
        /// Sets and gets the TotalHoursOfClasses property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string TotalHoursDictated
        {
            get
            {
                return _totalHoursDictated;
            }
            set
            {
                Set(() => TotalHoursDictated, ref _totalHoursDictated, value);
            }
        }
    }
}
```

CourseResumeViewModel.cs

```
using System;
using System.Windows.Media;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseResumeViewModel : ViewModel
    {
        public string YearInCareer { get; set; }
        /// <summary>
        /// The <see cref="Title" /> property's name.
        /// </summary>
        public const string TitlePropertyName = "Title";

        private string _title = string.Empty;
```

```
/// <summary>
/// Sets and gets the Title property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Title
{
    get
    {
        return _title;
    }
    set
    {
        Set(TitlePropertyName, ref _title, value);
    }
}

/// <summary>
/// The <see cref="Professor" /> property's name.
/// </summary>
public const string ProfessorPropertyName = "Professor";

private string _professor = string.Empty;

/// <summary>
/// Sets and gets the Professor property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Professor
{
    get
    {
        return _professor;
    }
    set
    {
        Set(ProfessorPropertyName, ref _professor, value);
    }
}
/// <summary>
/// The <see cref="Attendance" /> property's name.
/// </summary>
public const string AttendancePropertyName = "Attendance";

private double _attendance = 0;

/// <summary>
```

```
        /// Sets and gets the Attendance property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double Attendance
        {
            get
            {
                return _attendance;
            }
            set
            {
                Set(AttendancePropertyName, ref _attendance, value);
            }
        }

        public int AttendanceLevel { get; set; }

        public SolidColorBrush AttendanceState
        {
            get
            {
                if (AttendanceLevel == 3)
                    return new SolidColorBrush(Colors.Green);
                return AttendanceLevel == 2 ? new SolidColorBrush(Colors.Yellow) : new
SolidColorBrush(Colors.Red);
            }
        }

        /// <summary>
        /// The <see cref="Code" /> property's name.
        /// </summary>
        public const string CodePropertyName = "Code";

        private int _code = 0;

        /// <summary>
        /// Sets and gets the Code property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public int Code
        {
            get
            {
                return _code;
            }
            set
            {
```

```
        Set(CodePropertyName, ref _code, value);
    }
}

/// <summary>
/// The <see cref="StudentCode" /> property's name.
/// </summary>
public const string StudentCodePropertyName = "StudentCode";

private int _studentCode = 0;

/// <summary>
/// Sets and gets the StudentCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int StudentCode
{
    get
    {
        return _studentCode;
    }
    set
    {
        Set(() => StudentCode, ref _studentCode, value);
    }
}

/// <summary>
/// The <see cref="CareerCode" /> property's name.
/// </summary>
public const string CareerCodePropertyName = "CareerCode";

private int _careerCode = 0;

/// <summary>
/// Sets and gets the CareerCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int CareerCode
{
    get
    {
        return _careerCode;
    }
    set
    {
        Set(() => CareerCode, ref _careerCode, value);
```

```
            }
        }

        private RelayCommand _coursePageDetails;
        public RelayCommand CoursePageDetails
        {
            get
            {
                return _coursePageDetails
                    ?? (_coursePageDetails = new RelayCommand(() =>
NavigationService.NavigateTo(new
Uri(String.Format("/View/CourseView.xaml?CareerCode={0}&StudentCode={1}&CourseCo
de={2}",

                                                                CareerCode,

                                                                StudentCode,

                                                                Code),

                                                    UriKind.Relative)

                        )

                                        ));
            }
        }

    }
}
```

CourseStudentSituationViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseStudentSituationViewModel : ViewModel
    {
        /// <summary>
        /// The <see cref="PartialExamResult" /> property's name.
        /// </summary>
```

```
        public const string PartialExamResultPropertyName = "PartialExamResult";

        private double _partialExamResult = 0;

        /// <summary>
        /// Sets and gets the PartialExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double PartialExamResult
        {
            get
            {
                return _partialExamResult;
            }
            set
            {
                Set(PartialExamResultPropertyName, ref _partialExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="RecuperationExamResult" /> property's name.
        /// </summary>
        public const string RecuperationExamResultPropertyName =
"RecuperationExamResult";

        private double _recuperationExamResult = 0;

        /// <summary>
        /// Sets and gets the RecuperationExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double RecuperationExamResult
        {
            get
            {
                return _recuperationExamResult;
            }
            set
            {
                Set(() => RecuperationExamResult, ref _recuperationExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="FinalExamResult" /> property's name.
        /// </summary>
```

```
        public const string FinalExamResultPropertyName = "FinalExamResult";

        private double _finalExamResult = 0;

        /// <summary>
        /// Sets and gets the FinalExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double FinalExamResult
        {
            get
            {
                return _finalExamResult;
            }
            set
            {
                Set(() => FinalExamResult, ref _finalExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="PracticWorkExamResult" /> property's name.
        /// </summary>
        public const string PracticWorkExamResultPropertyName =
"PracticWorkExamResult";

        private double _practicWorkExamResult = 0;

        /// <summary>
        /// Sets and gets the PracticWorkExamResult property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double PracticWorkExamResult
        {
            get
            {
                return _practicWorkExamResult;
            }
            set
            {
                Set(() => PracticWorkExamResult, ref _practicWorkExamResult, value);
            }
        }

        /// <summary>
        /// The <see cref="TotalAssistedHours" /> property's name.
        /// </summary>
```

```
        public const string TotalAssistedHoursPropertyName = "TotalAssistedHours";

        private double _totalAssistedHours = 0;

        /// <summary>
        /// Sets and gets the TotalAssistedHours property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public double TotalAssistedHours
        {
            get
            {
                return _totalAssistedHours;
            }
            set
            {
                Set(() => TotalAssistedHours, ref _totalAssistedHours, value);
            }
        }

        /// <summary>
        /// The <see cref="ActualAssistedPercent" /> property's name.
        /// </summary>
        public const string ActualAssistedPercentPropertyName = "ActualAssistedPercent";

        private string _actualAssistedPercent;

        /// <summary>
        /// Sets and gets the ActualAssistedPercent property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string ActualAssistedPercent
        {
            get
            {
                return _actualAssistedPercent;
            }
            set
            {
                Set(() => ActualAssistedPercent, ref _actualAssistedPercent, value);
            }
        }
    }
}
```

CoursesViewModel.cs

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Windows;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Client.Model;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
   public class CoursesViewModel : ViewModel
   {
      private readonly IStudentRepository _studentRepository;

      public CoursesViewModel(IStudentRepository studentRepository)
      {
         _studentRepository = studentRepository;
         Courses = new
ObservableCollection<GroupCourses<CourseResumeViewModel>>();

         BuildCourseResume(_studentRepository.GetStudent());
      }

      private void BuildCourseResume(Student student)
      {
         if (student == null) return;

         var courses = student.Courses;
         if (courses == null) return;
         Deployment.Current.Dispatcher.BeginInvoke(() =>
            {
               var courseResumeList = new List<CourseResumeViewModel>();
               courses.Where(c => c.IsCurretCourse)
                     .ToList()
                     .ForEach(c => courseResumeList.Add(new CourseResumeViewModel
                        {
                           Title = c.Course.Title,
                           Code = c.Course.Code,
                           Professor = c.Course.Professor,
                           Attendance = c.Attendance(),
                           AttendanceLevel = c.AttendanceState,
                           StudentCode = student.StudentCode,
                           CareerCode = student.CareerCode,
                           YearInCareer = c.Course.YearOfCareer.ToString()
                        }));
```

```
                GroupCourses<CourseResumeViewModel>.CreateGroups(courseResumeList)
                                .ToList()
                                .ForEach(g => Courses.Add(g));
            });
        }


        public ObservableCollection<GroupCourses<CourseResumeViewModel>> Courses {
    get; private set; }

        public override void OnNavigateTo(IDictionary<string, string> queryString)
        {
            Courses.Clear();
            BuildCourseResume(_studentRepository.GetStudent());
        }
        public override void OnNavigateFrom(IDictionary<string, string> queryString)
        {
            base.OnNavigateFrom(queryString);
        }
    }
}
```

CourseViewModel.cs

```
using System.Windows;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;
using System;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class CourseViewModel : ViewModel
    {
        private readonly IStudentCourseRepository _repository;
        private readonly ISyncronizationService _syncService;
        private readonly IStudentInformationManager _studentInformationManager;

        public CourseViewModel(IStudentCourseRepository repository,
    ISyncronizationService syncService, IStudentInformationManager
    studentInformationManager)
        {
            _repository = repository;
            _syncService = syncService;
            _studentInformationManager = studentInformationManager;
        }
```

```
/// <summary>
/// The <see cref="Title" /> property's name.
/// </summary>
public const string TitlePropertyName = "Title";
private string _title = string.Empty;

/// <summary>
/// Sets and gets the Title property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string Title
{
    get
    {
        return _title;
    }
    set
    {
        Set(() => Title, ref _title, value);
    }
}

#region Description PivotItem
/// <summary>
/// The <see cref="CourseDescription" /> property's name.
/// </summary>
public const string CourseDescriptionPropertyName = "CourseDescription";

private CourseDescriptionViewModel _myProperty = null;

/// <summary>
/// Sets and gets the CourseDescription property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public CourseDescriptionViewModel CourseDescriptionViewModel
{
    get
    {
        return _myProperty;
    }

    set
    {
        if (_myProperty == value)
        {
            return;
```

```
        }

        RaisePropertyChanging(() => CourseDescriptionViewModel);
        _myProperty = value;
        RaisePropertyChanged(() => CourseDescriptionViewModel);
    }
}
#endregion

#region My Situation PivotItem
/// <summary>
/// The <see cref="CourseSituation" /> property's name.
/// </summary>
public const string CourseSituationPropertyName = "CourseSituation";

private CourseStudentSituationViewModel _courseStudent = null;

/// <summary>
/// Sets and gets the CourseSituation property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public CourseStudentSituationViewModel CourseStudentSituationViewModel
{
    get
    {
        return _courseStudent;
    }

    set
    {
        if (_courseStudent == value)
        {
            return;
        }

        RaisePropertyChanging(() => CourseStudentSituationViewModel);
        _courseStudent = value;
        RaisePropertyChanged(() => CourseStudentSituationViewModel);
    }
}


#endregion

public int Code { get; set; }
public int StudentCode { get; private set; }
public int CareerCode { get; set; }
```

```
        public override void OnNavigateTo(System.Collections.Generic.IDictionary<string,
string> queryString)
        {
            string code;
            string studentCode;
            string careerCode;

            if (queryString.TryGetValue("StudentCode", out studentCode))
                StudentCode = int.Parse(studentCode);
            if (queryString.TryGetValue("CareerCode", out careerCode))
                CareerCode = int.Parse(careerCode);

            if (queryString.TryGetValue("CourseCode", out code))
            {
                var course = _repository.GetCourse(code);
                Code = int.Parse(code);
                CourseDescriptionViewModel = new CourseDescriptionViewModel
                {
                    Description = course.Course.Description,
                    FinalExamDate = course.Course.FinalExamDate != null ?
((DateTime)course.Course.FinalExamDate).ToShortDateString() : string.Empty,
                    PartialExamDate = course.Course.PartialExamDate != null ?
((DateTime)course.Course.PartialExamDate).ToShortDateString() : string.Empty,
                    PracticWorkExamDate = course.Course.PracticWorkPresentationDate != null ?
((DateTime)course.Course.PracticWorkPresentationDate).ToShortDateString() :
string.Empty,
                    Professor = course.Course.Professor,
                    RecuperationExamDate = course.Course.RecuperationExameDate != null ?
((DateTime)course.Course.RecuperationExameDate).ToShortDateString() : string.Empty,
                    Scheduled = course.Course.Scheduled,
                    TotalHoursOfClasses = String.Format("{0} horas catedra",
course.Course.TotalHoursOfClasses),
                    TotalHoursInAWeek = String.Format("{0} horas catedra",
course.Course.TotalHoursInAWeek),
                    TotalHoursDictated = String.Format("{0} horas catedra",
course.Course.TotalHoursDictated)
                };
                CourseStudentSituationViewModel = new CourseStudentSituationViewModel
                {
                    ActualAssistedPercent = string.Format("{0}%",course.Attendance()),
                    FinalExamResult = course.FinalExamResult,
                    PartialExamResult = course.PartialExamResult,
                    RecuperationExamResult = course.RecuperationExamResult,
                    PracticWorkExamResult = course.PracticWorkResult,
                    TotalAssistedHours = course.TotalHoursAssisted
```

```
            };
            Title = course.Course.Title;
            CanCheckForFinalExam = EvaluateStudentCanRegisterForFinalExam(course);
            SyncCourseInfo.Execute(null);
        }
    }

    private bool EvaluateStudentCanRegisterForFinalExam(StudentCourse course)
    {
        if (course.FinalExamInscripted)
            return false;
        else
        {
            if (course.AttendanceState != 1)
            {
                if (course.FinalExamResult == 0)
                    if (course.PartialExamResult >= 4 || course.RecuperationExamResult >= 4)
                        return true;
                    else
                        return false;
                else
                    return false;
            }
            else
                return false;
        }
    }

    public override void OnNavigateFrom(System.Collections.Generic.IDictionary<string,
string> queryString)
    {
        var course = _repository.GetCourse(Code.ToString());

        var command = new RelayCommand(() =>
_studentInformationManager.AddCourseExamsDatesReminders(course).Subscribe(
                x => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; }),
                ex => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false;
}),
                () => Deployment.Current.Dispatcher.BeginInvoke(() =>
                {
                    Updating = false;
                    base.OnNavigateFrom(queryString);
                })));

        command.Execute(null);
    }
```

```
        private RelayCommand _syncCourseInfo;

        /// <summary>
        /// Gets the MyCommand.
        /// </summary>
        public RelayCommand SyncCourseInfo
        {
            get
            {
                Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; });
                return _syncCourseInfo
                    ?? (_syncCourseInfo = new RelayCommand(() =>
_syncService.SyncCourse(Code, StudentCode.ToString(),
CareerCode.ToString()).Subscribe(
                        x => Deployment.Current.Dispatcher.BeginInvoke(() =>
                            {
                                //DescriptionViewModel
                                CourseDescriptionViewModel.Description = x.Course.Description;
                                CourseDescriptionViewModel.FinalExamDate =
x.Course.FinalExamDate != null ?
((DateTime)x.Course.FinalExamDate).ToShortDateString() : string.Empty;
                                CourseDescriptionViewModel.PartialExamDate =
x.Course.PartialExamDate != null ?
((DateTime)x.Course.PartialExamDate).ToShortDateString() : string.Empty;
                                CourseDescriptionViewModel.Professor = x.Course.Professor;
                                CourseDescriptionViewModel.PracticWorkExamDate =
x.Course.PracticWorkPresentationDate != null ?
((DateTime)x.Course.PracticWorkPresentationDate).ToShortDateString() : string.Empty;
                                CourseDescriptionViewModel.Scheduled = x.Course.Scheduled;
                                CourseDescriptionViewModel.RecuperationExamDate =
x.Course.RecuperationExameDate != null ?
((DateTime)x.Course.RecuperationExameDate).ToShortDateString() : string.Empty;

                                //MySituationViewModel
                                CourseStudentSituationViewModel.ActualAssistedPercent =
string.Format("{0}%",x.Attendance());
                                CourseStudentSituationViewModel.FinalExamResult =
x.FinalExamResult;
                                CourseStudentSituationViewModel.PartialExamResult =
x.PartialExamResult;
                                CourseStudentSituationViewModel.PracticWorkExamResult =
x.PracticWorkResult;
                                CourseStudentSituationViewModel.RecuperationExamResult =
x.RecuperationExamResult;
                                CourseStudentSituationViewModel.TotalAssistedHours =
x.TotalHoursAssisted;
```

```
                        //CanRegisterForFinalExam
                        CanCheckForFinalExam =
EvaluateStudentCanRegisterForFinalExam(x);

                        _repository.AddCourse(x);
                    }),
                (Exception ex) => Deployment.Current.Dispatcher.BeginInvoke(() => {
Updating = false; }),
                () => Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false;
})));
        }
    }

    private RelayCommand _checkForFinalExam;

    /// <summary>
    /// Gets the CheckForFinalExam.
    /// </summary>
    public RelayCommand CheckForFinalExam
    {
        get
        {
            return _checkForFinalExam
                ?? (_checkForFinalExam = new RelayCommand(() =>
_syncService.CheckStudentForFinalExamOfCourse(StudentCode.ToString(),
Code.ToString()).Subscribe(
                                        x =>
Deployment.Current.Dispatcher.BeginInvoke(() =>
                                        {
                                            MessageBox.Show(x.Message);
                                            //x.Result == TaskResult.Success ? "La
inscripción se ha realizado con exito" :
                                            //                "Ha ocurrido
un problema al realizar la " +
                                            //                "inscripción.
Por favor pruebe en unos minutos");
                                        }),
                                        (Exception ex) =>
Deployment.Current.Dispatcher.BeginInvoke(() => MessageBox.Show("Se ha producido un
error. Dirijase a bedelia por favor")),
                                        () =>
Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false; })));
        }
    }
```

```
        public bool CanCheckForFinalExam { get; set; }
    }
}
```

MainViewModel.cs

```
using System;
using GalaSoft.MvvmLight.Command;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class MainViewModel : ViewModel
    {
        private readonly IStudentRepository _studentRepository;

        public MainViewModel(IStudentRepository studentRepository)
        {
            _studentRepository = studentRepository;

            var student = _studentRepository.GetStudent();
            if (student != null)
            {
                StudentCode = student.StudentCode;
                CareerCode = student.CareerCode.ToString();
            }

        }

        public string CareerCode { get; set; }
        /// <summary>
        /// The <see cref="StudentCode" /> property's name.
        /// </summary>
        public const string StudentCodePropertyName = "StudentCode";

        private int _studentCode = 0;
        /// <summary>
        /// Sets and gets the StudentCode property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public int StudentCode
        {
            get
            {
                return _studentCode;
            }
```

```
            set
            {
                if (_studentCode == value)
                {
                    return;
                }

                RaisePropertyChanging(() => StudentCode);
                _studentCode = value;
                RaisePropertyChanged(() => StudentCode);
            }
        }

        private RelayCommand _configurationPageNavigation;

        /// <summary>
        /// Gets the MyCommand.
        /// </summary>
        public RelayCommand NavigateToSettingsView
        {
            get
            {
                return _configurationPageNavigation
                    ?? (_configurationPageNavigation = new RelayCommand(() =>
NavigationService.NavigateTo(new
Uri(String.Format("/View/SettingsView.xaml?StudentCode={0}&CareerCode={1}",
StudentCode, CareerCode),UriKind.Relative))));
            }
        }

        private RelayCommand _navigateToAllCoursesView;

        /// <summary>
        /// Gets the MyCommand.
        /// </summary>
        public RelayCommand NavigateToAllCoursesView
        {
            get
            {
                return _navigateToAllCoursesView
                    ?? (_navigateToAllCoursesView = new RelayCommand(
                                () =>
                                    {
                                        NavigationService.NavigateTo(new
Uri("/View/AllCoursesView.xaml",
                                                                     UriKind.Relative));
                                    }));
```

```
            }
        }

        public override void OnNavigateTo(System.Collections.Generic.IDictionary<string,
string> queryString)
        {
            string careerCode;
            string studentCode;
            string courseCode;
            queryString.TryGetValue("StudentCode", out studentCode);
            queryString.TryGetValue("CareerCode", out careerCode);
            queryString.TryGetValue("CourseCode", out courseCode);

            if (string.IsNullOrEmpty(careerCode) || string.IsNullOrEmpty(careerCode) ||
string.IsNullOrEmpty(careerCode))
            {
                base.OnNavigateTo(queryString);
            }
            else
            {

                NavigationService.NavigateTo(
                    new Uri(String.Format(

"/Views/CourseView.xaml?CareerCode={0}&amp;StudentCode={1}&amp;CourseCode={2}"
,
                        careerCode,
                        studentCode,
                        courseCode)));
            }
        }
    }
}
```

SettingsViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Windows;
using GalaSoft.MvvmLight.Command;
using Microsoft.Phone.Notification;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class SettingsViewModel : ViewModel
```

```
    {
        private readonly IStudentRepository _studentRepository;
        private readonly ISyncronizationService _syncronizationService;

        public SettingsViewModel(IStudentRepository studentRepository,
ISyncronizationService syncronizationService)
        {
            _studentRepository = studentRepository;
            _syncronizationService = syncronizationService;
            var student = _studentRepository.GetStudent();

            if (student != null)
                if (!string.IsNullOrEmpty(student.DeviceUri))
                    _isChecked = 1;
        }

        /// <summary>
        /// The <see cref="StudentCode" /> property's name.
        /// </summary>
        public const string StudentCodePropertyName = "StudentCode";
        private string _studentCode = string.Empty;
        /// <summary>
        /// Sets and gets the StudentCode property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public string StudentCode
        {
            get
            {
                return _studentCode;
            }

            set
            {
                if (_studentCode == value)
                {
                    return;
                }

                RaisePropertyChanging(StudentCodePropertyName);
                _studentCode = value;
                RaisePropertyChanged(StudentCodePropertyName);
            }
        }

        /// <summary>
        /// The <see cref="IsChecked" /> property's name.
```

```
/// </summary>
public const string IsCheckedPropertyName = "IsChecked";
private int? _isChecked = null;
/// <summary>
/// Sets and gets the IsChecked property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public int? IsChecked
{
    get
    {
        return _isChecked;
    }

    set
    {
        if (_isChecked == value)
        {
            return;
        }

        RaisePropertyChanging(IsCheckedPropertyName);
        _isChecked = value;
        RaisePropertyChanged(IsCheckedPropertyName);
    }
}

/// <summary>
/// The <see cref="CareerCode" /> property's name.
/// </summary>
public const string CareerCodePropertyName = "CareerCode";
private string _careerCode = string.Empty;
/// <summary>
/// Sets and gets the CareerCode property.
/// Changes to that property's value raise the PropertyChanged event.
/// </summary>
public string CareerCode
{
    get
    {
        return _careerCode;
    }

    set
    {
        if (_careerCode == value)
        {
```

```
            return;
        }

        RaisePropertyChanging(CareerCodePropertyName);
        _careerCode = value;
        RaisePropertyChanged(CareerCodePropertyName);
    }
}

private RelayCommand _saveChangesInSettings;
/// <summary>
/// Gets the MyCommand.
/// </summary>
public RelayCommand SaveChangesInSettings
{
    get
    {
        return _saveChangesInSettings
            ?? (_saveChangesInSettings = new RelayCommand(
                                () =>
                                {
                                    if (IsChecked != 1) return;
                                    GetNotificationUri();
                                }));
    }
}

private void GetNotificationUri()
{
    const string channelName = "SiaMobileWindowsPhoneApplication";

    var pushChannel = HttpNotificationChannel.Find(channelName);

    // If the channel was not found, then create a new connection to the push service.
    if (pushChannel == null)
    {
        pushChannel = new HttpNotificationChannel(channelName);

        // Register for all the events before attempting to open the channel.
        pushChannel.ChannelUriUpdated += PushChannel_ChannelUriUpdated;
        pushChannel.ErrorOccurred += PushChannel_ErrorOccurred;

        pushChannel.Open();
        // Bind this new channel for Tile events.
        pushChannel.BindToShellTile();
        pushChannel.BindToShellToast();
    }
```

```
            else
            {
                // The channel was already open, so just register for all the events.
                pushChannel.ChannelUriUpdated += PushChannel_ChannelUriUpdated;
                pushChannel.ErrorOccurred += PushChannel_ErrorOccurred;

                if (!pushChannel.IsShellTileBound) pushChannel.BindToShellTile();
                if (!pushChannel.IsShellToastBound) pushChannel.BindToShellToast();
            }
        }

        void PushChannel_ChannelUriUpdated(object sender,
NotificationChannelUriEventArgs e)
        {
            Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = true; });
            _syncronizationService.RegisterNotificationChannelForStudent(StudentCode,
CareerCode, e.ChannelUri.ToString()).Subscribe(
                                        (TaskSummary x) =>
                                        {
                                            if (x.Result == TaskResult.Failed)
                                            {
                                                Deployment.Current.Dispatcher.BeginInvoke(() =>
MessageBox.Show("Se ha producido un error al tratar de guardar la configuración"));
                                            }
                                        },
                                        (Exception ex) =>
Deployment.Current.Dispatcher.BeginInvoke(() => { Updating = false; }),
                                        () => Deployment.Current.Dispatcher.BeginInvoke(() => {
Updating = false; }));
        }

        void PushChannel_ErrorOccurred(object sender, NotificationChannelErrorEventArgs
e)
        {
            Deployment.Current.Dispatcher.BeginInvoke(() => MessageBox.Show("Se ha
producido un error al tratar de guardar la configuración"));
        }



        public override void OnNavigateTo(IDictionary<string, string> queryString)
        {
            string studentCode;
            string courseCode;

            if (queryString.TryGetValue("StudentCode", out studentCode))
            {
```

```
                _studentCode = studentCode;
            }

            if (queryString.TryGetValue("CareerCode", out courseCode))
            {
                _careerCode = courseCode;
            }

            base.OnNavigateTo(queryString);
        }
    }
}
```

ViewModel.cs

```
using System.Collections.Generic;
using GalaSoft.MvvmLight;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;
using GalaSoft.MvvmLight.Ioc;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public abstract class ViewModel : ViewModelBase
    {
        protected readonly INavigationService NavigationService;

        protected ViewModel()
        {
            NavigationService = SimpleIoc.Default.GetInstance<INavigationService>();
        }

        /// <summary>
        /// The <see cref="Updating" /> property's name.
        /// </summary>
        public const string UpdatingPropertyName = "Updating";

        private bool _updating = false;

        /// <summary>
        /// Sets and gets the Updating property.
        /// Changes to that property's value raise the PropertyChanged event.
        /// </summary>
        public bool Updating
        {
            get
            {
                return _updating;
```

```
            }

            set
            {
                if (_updating == value)
                {
                    return;
                }

                RaisePropertyChanging(UpdatingPropertyName);
                _updating = value;
                RaisePropertyChanged(UpdatingPropertyName);
            }
        }

        public virtual void OnNavigateTo(IDictionary<string, string> queryString)
        {
        }

        public virtual void OnNavigateFrom(IDictionary<string, string> queryString)
        {
        }
    }
}
```

ViewModelLocator.cs

```
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;
using TesinaMobileCloud.Phone.Client.ViewServices;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;
using TesinaMobileCloud.Phone.Data.Repositories;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewModel
{
    public class ViewModelLocator
    {
        static ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            // Create run time view services and models
            SimpleIoc.Default.Register<IStudentCourseRepository,
StudentCourseRepository>();
```

```
            SimpleIoc.Default.Register<IStudentRepository, StudentRepository>();
            SimpleIoc.Default.Register<IScheduleActionClient, ScheduleActionClient>();
            SimpleIoc.Default.Register<IScheduleActionService,
ScheduleActionServiceAdapter>();
            SimpleIoc.Default.Register<INavigationService, NavigationService>();
            SimpleIoc.Default.Register<ISyncronizationService, SyncronizationService>();
            SimpleIoc.Default.Register<ICloudService, CloudService>();
            SimpleIoc.Default.Register<IStudentInformationManager,
StudentInformationManager>();


            SimpleIoc.Default.Register<MainViewModel>();
            SimpleIoc.Default.Register<SettingsViewModel>();
            SimpleIoc.Default.Register<CoursesViewModel>();
            SimpleIoc.Default.Register<CourseResumeViewModel>();
            SimpleIoc.Default.Register<CourseViewModel>();
            SimpleIoc.Default.Register<CourseStudentSituationViewModel>();
            SimpleIoc.Default.Register<CourseDescriptionViewModel>();
            SimpleIoc.Default.Register<AllCoursesViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }
        public CoursesViewModel Courses
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CoursesViewModel>();
            }
        }
        public CourseResumeViewModel CourseResume
        {
            get
            {
                return ServiceLocator.Current.GetInstance<CourseResumeViewModel>();
            }

        }
        public CourseViewModel Course
        {
            get
            {
```

```
            return ServiceLocator.Current.GetInstance<CourseViewModel>();
        }

    }
    public CourseDescriptionViewModel CourseDescription
    {
        get
        {
            return ServiceLocator.Current.GetInstance<CourseDescriptionViewModel>();
        }

    }
    public CourseStudentSituationViewModel CourseStudentSituation
    {
        get
        {
            return
ServiceLocator.Current.GetInstance<CourseStudentSituationViewModel>();
        }

    }
    public IScheduleActionClient ScheduleActions
    {
        get
        {
            return ServiceLocator.Current.GetInstance<IScheduleActionClient>();
        }
    }

    public SettingsViewModel Settings
    {
        get { return ServiceLocator.Current.GetInstance<SettingsViewModel>(); }
    }

    public AllCoursesViewModel AllCourses
    {
        get { return ServiceLocator.Current.GetInstance<AllCoursesViewModel>(); }
    }

    public static void Cleanup()
    {
        // TODO Clear the ViewModels
    }
  }
}
```

ViewServices

NavigationService.cs

```
using Microsoft.Phone.Controls;
using System;
using System.Windows;
using System.Windows.Navigation;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.ViewServices
{
    public class NavigationService : INavigationService
    {
        private PhoneApplicationFrame _mainFrame;

        public event NavigatingCancelEventHandler Navigating;

        public void NavigateTo(Uri pageUri)
        {
            if (InitializatedFrame())
            {
                _mainFrame.Navigate(pageUri);
            }
        }

        public void GoBack()
        {
            if (InitializatedFrame() && _mainFrame.CanGoBack)
            {
                _mainFrame.GoBack();
            }
        }

        private bool InitializatedFrame()
        {
            if (_mainFrame != null)
                return true;

            _mainFrame = Application.Current.RootVisual as PhoneApplicationFrame;

            if (_mainFrame != null)
            {
                _mainFrame.Navigating += (s, e) =>
                {
                    if (Navigating != null)
                    {
                        Navigating(s, e);
```

```
                }
            };

            return true;
        }

        return false;
    }
  }
}
```

Interfaces

INavigationService.cs

```
using System;
using System.Windows.Navigation;

namespace TesinaMobileCloud.Phone.Client.ViewServices.Interfaces
{
    public interface INavigationService
    {
        event NavigatingCancelEventHandler Navigating;
        void NavigateTo(Uri pageUri);
        void GoBack();
    }

}
```

TesinaMobileCloud.Phone.Client.Test

App.xaml

```
<?xml version="1.0" encoding="utf-8"?>
<Application x:Class="TesinaMobileCloud.Phone.Client.Test.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone" xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test.ViewModel" mc:Ignorable="d">
  <!--Application Resources-->
  <Application.Resources>
    <ResourceDictionary>
      <vm:ViewModelLocator x:Key="Locator" d:IsDataSource="True" />
      <ResourceDictionary.MergedDictionaries></ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
  <Application.ApplicationLifetimeObjects>
```

```
    <!--Required object that handles lifetime events for the application-->
    <shell:PhoneApplicationService Launching="Application_Launching"
Closing="Application_Closing" Activated="Application_Activated"
Deactivated="Application_Deactivated" />
  </Application.ApplicationLifetimeObjects>
</Application>
```

App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class App : Application
    {
        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();
```

```
            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Disable the application idle detection by setting the UserIdleDetectionMode
property of the
                // application's PhoneApplicationService object to Disabled.
                // Caution:- Use this under debug mode only. Application that disables user idle
detection will continue to run
                // and consume battery power when the user is not using the phone.
                PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
            }

        }

        // Code to execute when the application is launching (eg, from Start)
        // This code will not execute when the application is reactivated
        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
        }

        // Code to execute when the application is activated (brought to foreground)
        // This code will not execute when the application is first launched
        private void Application_Activated(object sender, ActivatedEventArgs e)
        {
        }

        // Code to execute when the application is deactivated (sent to background)
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEventArgs e)
        {
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
```

```
        {
        }

        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender, NavigationFailedEventArgs
e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // A navigation has failed; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender,
ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows the splash
            // screen to remain active until the application is ready to render.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }
```

```
        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, NavigationEventArgs
    e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="TesinaMobileCloud.Phone.Client.Test.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>
```

```
        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Testing;

namespace TesinaMobileCloud.Phone.Client.Test
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            this.Content = UnitTestSystem.CreateTestPage();
        }
    }
}
```

UnitTest

CoursesViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.ObjectModel;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CoursesViewModelFixture
```

```
    {
        readonly IStudentRepository _studentRepository = new MockStudentRepository();

        [TestMethod]
        public void CoursesViewModel()
        {
            var sut = new CoursesViewModel(_studentRepository);

            Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
            Assert.IsInstanceOfType(sut.Courses,
typeof(ObservableCollection<CourseResumeViewModel>));
        }

        [TestMethod]
        public void CoursesViewModelFillCoursesList()
        {
            var sut = new CoursesViewModel(_studentRepository);

            Assert.IsNotNull(sut.Courses);
            Assert.AreEqual(1, sut.Courses.Count);
        }
    }
}
```

CourseViewModelFixture.cs

```
using GalaSoft.MvvmLight;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Windows.Media;
using TesinaMobileCloud.Phone.Client.ViewModel;
using TesinaMobileCloud.Phone.Client.ViewServices.Interfaces;

namespace TesinaMobileCloud.Phone.Client.Test.UnitTest
{
    [TestClass]
    public class CourseViewModelFixture
    {

        [TestMethod]
        public void CourseViewModelWithParams()
        {
            var sut = new CourseResumeViewModel
            {
                Code = 124,
                Title = "Title",
                Professor = "Prof",
                //YearOfCareer = 1,
```

```
            Attendance = 1
        };

        Assert.IsInstanceOfType(sut, typeof(ViewModelBase));
        Assert.IsNotNull(sut);
        Assert.AreEqual(124, sut.Code);
        Assert.AreEqual("Title", sut.Title);
        //Assert.AreEqual(1, sut.YearOfCareer);
        Assert.AreEqual("Prof", sut.Professor);
        Assert.AreEqual(1, sut.Attendance);
        Assert.IsInstanceOfType(sut.AttendanceState, typeof(SolidColorBrush));
    }

    [TestMethod]
    public void CourseViewModelWithOutParams()
    {
        var sut = new CourseResumeViewModel();
        Assert.IsNotNull(sut);
    }
  }
}
```

ViewModel

MainViewModel.cs

```
using GalaSoft.MvvmLight;

namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
    /// <summary>
    /// This class contains properties that the main View can data bind to.
    /// <para>
    /// Use the <strong>mvvminpc</strong> snippet to add bindable properties to this
ViewModel.
    /// </para>
    /// <para>
    /// You can also use Blend to data bind with the tool's support.
    /// </para>
    /// <para>
    /// See http://www.galasoft.ch/mvvm
    /// </para>
    /// </summary>
    public class MainViewModel : ViewModelBase
    {
        /// <summary>
```

```
        /// Initializes a new instance of the MainViewModel class.
        /// </summary>
        public MainViewModel()
        {
            ////if (IsInDesignMode)
            ////{
            ////    // Code runs in Blend --> create design time data.
            ////}
            ////else
            ////{
            ////    // Code runs "for real"
            ////}
        }
    }
}
```

ViewModelLocator.cs

```
/*
  In App.xaml:
  <Application.Resources>
      <vm:ViewModelLocator xmlns:vm="clr-
namespace:TesinaMobileCloud.Phone.Client.Test"
                           x:Key="Locator" />
  </Application.Resources>

  In the View:
  DataContext="{Binding Source={StaticResource Locator}, Path=ViewModelName}"

  You can also use Blend to do all this with the tool's support.
  See http://www.galasoft.ch/mvvm
*/

using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Ioc;
using Microsoft.Practices.ServiceLocation;

namespace TesinaMobileCloud.Phone.Client.Test.ViewModel
{
    /// <summary>
    /// This class contains static references to all the view models in the
    /// application and provides an entry point for the bindings.
    /// </summary>
    public class ViewModelLocator
    {
        /// <summary>
        /// Initializes a new instance of the ViewModelLocator class.
        /// </summary>
```

```
        public ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);

            ////if (ViewModelBase.IsInDesignModeStatic)
            ////{
            ////    // Create design time view services and models
            ////    SimpleIoc.Default.Register<IDataService, DesignDataService>();
            ////}
            ////else
            ////{
            ////    // Create run time view services and models
            ////    SimpleIoc.Default.Register<IDataService, DataService>();
            ////}

            SimpleIoc.Default.Register<MainViewModel>();
        }

        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

TesinaMobileCloud.Phone.Data

Repositories

MockStudentCourseRepository.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class MockStudentCourseRepository : IStudentCourseRepository
    {
        private static List<StudentCourse> _courses;
```

```
        public MockStudentCourseRepository()
        {
            _courses = new List<StudentCourse>
            {
                new StudentCourse
                {
                    Course = new Course
                    {
                        Code = 1,
                        Title = "Programación I",
                        Professor = "Ing. Carlos Rodrigo",
                        Description = "Programacion I introduce los conceptos basicos de la
programacion, "+
                        "comenzado con el paradigma estructurado y luego introduciendo el
paradigma de objetos. "+
                        "El lenguaje de programacion a utilizar es Java.",
                        TotalHoursOfClasses = 60,
                        Scheduled = "Miercoles 17Hs",
                        FinalExamDate =new DateTime(2013,02,4,00,22,00),
                        PartialExamDate = new DateTime(2013,02,4,00,21,00),
                        PracticWorkPresentationDate = DateTime.MinValue,
                        RecuperationExameDate = DateTime.MinValue
                    },
                    FinalExamResult = 10,
                    PartialExamResult = 9,
                    PracticWorkResult = 9,
                    RecuperationExamResult = 0,
                    TotalHoursAssisted = 40
                }
            };
        }

        public IEnumerable<StudentCourse> GetCourses()
        {
            return _courses;
        }

        public void AddCourses(IEnumerable<StudentCourse> courses)
        {
            _courses.Clear();
            foreach (var course in courses)
                _courses.Add(course);
        }

        public void AddCourse(StudentCourse course)
        {
```

```
            throw new NotImplementedException();
        }

        public StudentCourse GetCourse(string code)
        {
            return new StudentCourse
            {
                Course = new Course
                {
                    Code = 1,
                    Title = "Programación I",
                    Professor = "Ing. Carlos Rodrigo",
                    Description = "Programacion I introduce los conceptos basicos de la
programacion, " +
                                  "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. " +
                                  "El lenguaje de programacion a utilizar es Java.",
                    TotalHoursOfClasses = 60,
                    Scheduled = "Miercoles 17Hs",
                    FinalExamDate = new DateTime(2013, 03, 6, 00, 28, 00),
                    PartialExamDate = new DateTime(2013, 03, 6, 00, 26, 00),
                    PracticWorkPresentationDate = DateTime.MinValue,
                    RecuperationExameDate = DateTime.MinValue
                },
                FinalExamResult = 10,
                PartialExamResult = 9,
                PracticWorkResult = 9,
                RecuperationExamResult = 0,
                TotalHoursAssisted = 40
            };
        }
    }

}
```

StudentCourseRepository.cs

```
using System.Collections.Generic;
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class StudentCourseRepository : IStudentCourseRepository
    {
        private const string studentcourseFormat = "StudentCourse{0}";
```

```
private readonly IsolatedStorageSettings _isolatedStorage;

public StudentCourseRepository()
{
    _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
}

public IEnumerable<StudentCourse> GetCourses()
{
    IEnumerable<StudentCourse> courses;
    _isolatedStorage.TryGetValue("StudentCourse", out courses);
    return courses;
}

public void AddCourses(IEnumerable<StudentCourse> courses)
{
    foreach (var course in courses)
    {
        AddCourseToIsolatedStorage(course);
    }
    _isolatedStorage.Save();
}

public void AddCourse(StudentCourse course)
{
    AddCourseToIsolatedStorage(course);
    _isolatedStorage.Save();
}

public StudentCourse GetCourse(string code)
{
    StudentCourse course;
    var courseCode = string.Format(studentcourseFormat, code);
    _isolatedStorage.TryGetValue(courseCode, out course);
    return course;
}

private void AddCourseToIsolatedStorage(StudentCourse course)
{
    StudentCourse studentCourse;
    var courseCode = string.Format(studentcourseFormat, course.Course.Code);
    if (_isolatedStorage.TryGetValue(courseCode, out studentCourse))
    {
        _isolatedStorage.Remove(courseCode);
    }
    _isolatedStorage.Add(courseCode, course);
}
```

```
    }
}
```

StudentRepository.cs

```
using System.IO.IsolatedStorage;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;

namespace TesinaMobileCloud.Phone.Data.Repositories
{
    public class StudentRepository : IStudentRepository
    {
        private readonly IsolatedStorageSettings _isolatedStorage;

        public StudentRepository()
        {
            _isolatedStorage = IsolatedStorageSettings.ApplicationSettings;
        }

        public void AddStudent(Student student)
        {
            Student studentToSave;
            if (_isolatedStorage.TryGetValue("Student", out studentToSave))
            {
                _isolatedStorage.Remove("Student");
            }
            _isolatedStorage.Add("Student", student);
            _isolatedStorage.Save();
        }

        public Student GetStudent()
        {
            Student student;
            _isolatedStorage.TryGetValue("Student", out student);
            return student;
        }
    }
}
```

Interfaces

IStudentCourseRepository.cs

```
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Repositories.Interfaces
{
    public interface IStudentCourseRepository
```

```
    {
        void AddCourses(IEnumerable<StudentCourse> courses);
        void AddCourse(StudentCourse course);
        StudentCourse GetCourse(string code);
    }
}
```

IStudentRepository.cs

```
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Repositories.Interfaces
{
    public interface IStudentRepository
    {
        void AddStudent(Student student);
        Student GetStudent();
    }

    public class MockStudentRepository : IStudentRepository
    {
        public void AddStudent(Student student)
        {
            throw new System.NotImplementedException();
        }

        public Student GetStudent()
        {
            return new Student
                {
                    Courses = new List<StudentCourse>
                                {
                                    new StudentCourse
                                    {
                                        Course = new Course
                                        {
                                            Code = 1,
                                            Title = "Programación I",
                                            Professor = "Ing. Carlos Rodrigo",
                                            Description = "Programacion I introduce los conceptos basicos
de la programacion, "+
                                            "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. "+
                                            "El lenguaje de programacion a utilizar es Java.",
                                            TotalHoursOfClasses = 60,
```

```
                                    Scheduled = "Miercoles 17Hs",
                                    FinalExamDate = new DateTime(2013,02,4,00,20,00),
                                    PartialExamDate = new DateTime(2013,02,4,00,21,00),
                                    PracticWorkPresentationDate = DateTime.MinValue,
                                    RecuperationExameDate = DateTime.MinValue
                                },
                                FinalExamResult = 10,
                                PartialExamResult = 9,
                                PracticWorkResult = 9,
                                RecuperationExamResult = 0,
                                TotalHoursAssisted = 40
                            }
                        },
                    CareerCode = 10004,
                    StudentCode = 502,
                    FirstName = "Carlos Sergio",
                    LastName = "Rodrigo"
                };
        }
    }
}
```

UnitTest1.cs

```
using System.Collections.Generic;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Data.Test
{
    [TestClass]
    public class StudentRepositoryFixture
    {
        [TestMethod]
        public void GetCourses()
        {
            var sut = new Data.Repositories.MockStudentCourseRepository();

            var result = sut.GetCourses();

            Assert.IsNotNull(result);
            Assert.IsInstanceOfType(result, typeof(IEnumerable<StudentCourse>));
        }

        [TestMethod]
        public void AddCourses()
```

```
        {
            var courses = new List<StudentCourse>
                        {
                            new StudentCourse
                            {
                                Course = new Course{
                                Title = "Programacion I", Professor = "Prof. Diana Ciccinelli"}
                            },
                            new StudentCourse
                            {
                                Course = new Course{
                                Title = "Programacion II", Professor = "Prof. Martin Cernadas"}
                            }
                        };
            var sut = new Data.Repositories.MockStudentCourseRepository();

            sut.AddCourses(courses);

            var result = sut.GetCourses();
            Assert.IsNotNull(result);
            Assert.AreEqual(2, result.Count());
        }
    }
}
```

TesinaMobileCloud.Phone.Services

CloudService.cs

```
using System;
using System.IO;
using System.Net;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Json;
using System.Text;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class CloudService : ICloudService
    {
        public IObservable<Student> GetAllCourses(Uri baseUri, string careerCode, string
studentCode, string password)
        {
```

```
        var relativeUri = String.Format("RetriveStudentByCodeAndCareer/{0}/{1}/{2}",
studentCode, careerCode, password);

        var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
        request.Method = "GET";
        request.Accept = "application/json";

        return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
            .Select(
                response =>
                {
                    using (var responseStream = response.GetResponseStream())
                    {
                        var serializer = new DataContractJsonSerializer(typeof(Student));
                        return serializer.ReadObject(responseStream) as Student;
                    }
                }
            );
    }

    public IObservable<StudentCourse> GetCourse(Uri baseUri, int code, string
studentCode, string careerCode)
    {
        var relativeUri = String.Format("RetriveStudentCourse/{0}/{1}/{2}", careerCode,
studentCode, code.ToString());
        var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
        request.Method = "GET";
        request.Accept = "application/json";

        return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
            .Select(
                response =>
                {
                    using (var responseStream = response.GetResponseStream())
                    {
                        var serializer = new DataContractJsonSerializer(typeof(StudentCourse));
                        return serializer.ReadObject(responseStream) as StudentCourse;
                    }
                }
            );
    }

    public IObservable<TaskSummary> RegisterNotificationChannelForStudent(Uri
baseUri, string studentCode, string careerCode, string uri)
    {
```

```
            var uriToSend = HttpUtility.UrlEncode(uri);
            var relativeUri =
String.Format("RegisterPushNotificationChannelForStudent?studentCode={0}&uri={1}",
studentCode, uriToSend);
            var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
            request.Method = "GET";
            request.Accept = "application/json";

            return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
                .Select(
                    response => new TaskSummary
                     {
                         Result = TaskResult.Success
                     })
                .Catch((Exception ex) => Observable.Return(new TaskSummary
                    {
                        Result = TaskResult.Failed
                    }));
        }


        public IObservable<TaskSummary> CheckStudentForFinalExamOfCourse(Uri
baseUri, string studentCode, string courseCode)
        {
            var relativeUri = String.Format("RegisterStudentForFinalExam/{0}/{1}",
studentCode, courseCode);
            var request = WebRequest.CreateHttp(new Uri(baseUri, relativeUri));
            request.Method = "GET";
            request.Accept = "application/json";

            return Observable.FromAsyncPattern<WebResponse>(request.BeginGetResponse,
request.EndGetResponse)()
                .Select(
                    response => new TaskSummary
                    {
                        Result = TaskResult.Success
                    })
                .Catch((Exception ex) => Observable.Return(new TaskSummary
                {
                    Result = TaskResult.Failed
                }));
        }
    }
}
```

ScheduleActionClient.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
   public class ScheduleActionClient : IScheduleActionClient
   {
      private const string PeriodicTaskName = "SyncCourses";
      private readonly IScheduleActionService _scheduleActionService;

      public ScheduleActionClient(IScheduleActionService scheduleActionService)
      {
         _scheduleActionService = scheduleActionService;
      }

      public void AddPeriodicTask()
      {
         var periodicTask = new PeriodicTask(PeriodicTaskName)
                        {
                           Description = "Syncronization Courses information Task"
                        };
         if (_scheduleActionService.Find(PeriodicTaskName) != null)
            _scheduleActionService.Remove(PeriodicTaskName);

         _scheduleActionService.Add(periodicTask);

#if DEBUG
         _scheduleActionService.LaunchForTest(PeriodicTaskName,
TimeSpan.FromMinutes(1));
#endif
      }

      public void RemovePeriodicTask()
      {
         if (_scheduleActionService.Find(PeriodicTaskName) != null)
            _scheduleActionService.Remove(PeriodicTaskName);
      }
   }
}
```

ScheduleActionServiceAdapter.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;
```

```
namespace TesinaMobileCloud.Phone.Services
{
    public class ScheduleActionServiceAdapter : IScheduleActionService
    {
        public ScheduledAction Find(string taskName)
        {
            return ScheduledActionService.Find(taskName);
        }

        public void Add(ScheduledAction action)
        {
            ScheduledActionService.Add(action);
        }

        public void Remove(string taskName)
        {
            ScheduledActionService.Remove(taskName);
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            ScheduledActionService.LaunchForTest(actionName, delay);
        }
    }
}
```

StudentInformationManager.cs

```
using System;
using Microsoft.Phone.Reactive;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Data.Repositories.Interfaces;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services
{
    public class StudentInformationManager : IStudentInformationManager
    {
        private readonly IStudentRepository _studentRepository;
        private readonly IStudentCourseRepository _studentCourseRepository;
        private readonly IScheduleActionService _scheduleActionService;

        public StudentInformationManager(IStudentRepository studentRepository,
    IStudentCourseRepository studentCourseRepository, IScheduleActionService
    scheduleActionService)
        {
            _studentRepository = studentRepository;
```

```
            _studentCourseRepository = studentCourseRepository;
            _scheduleActionService = scheduleActionService;
        }

        public string ProcessStudentInformation(Student student)
        {
            _studentRepository.AddStudent(student);
            foreach (var studentCourse in student.Courses)
            {
                ProcessStudentCourseInformation(studentCourse);
            }

            return string.Empty;
        }

        public void ProcessStudentCourseInformation(StudentCourse studentCourse)
        {
            _studentCourseRepository.AddCourse(studentCourse);
        }

        public IObservable<TaskSummary>
AddCourseExamsDatesReminders(StudentCourse studentCourse)
        {
            try
            {
                var courseTitle = studentCourse.Course.Title;
                if (studentCourse.Course.PartialExamDate != null)
                    AddRemainderForCourseExam(String.Format("{0}ExamParcial",
studentCourse.Course.Code),
                                              String.Format("Examen Parcial de {0}", courseTitle),
                                              String.Format("En d�a de ma�ana se dicta el Examen
Parcial de {0}", courseTitle),
                                              studentCourse.Course.PartialExamDate,
                                              24);
                if (studentCourse.Course.RecuperationExameDate != null)
                    AddRemainderForCourseExam(String.Format("{0}ExamRecuperatorio",
studentCourse.Course.Code),
                                              String.Format("Examen Recuperatorio de {0}", courseTitle),
                                              String.Format("En d�a de ma�ana se dicta el Examen
Recuperatorio de {0}", courseTitle),
                                              studentCourse.Course.RecuperationExameDate,
                                              24);
                if (studentCourse.Course.PracticWorkPresentationDate != null)

AddRemainderForCourseExam(String.Format("{0}PresentacionTrabajosPracticos",
studentCourse.Course.Code),
                                              String.Format("Presentaci�n de TPs {0}", courseTitle),
```

```
                                String.Format("En d�a de ma�ana se realiza la
Presentaci�n de TPs {0}", courseTitle),
                                studentCourse.Course.PracticWorkPresentationDate,
                                24);

            if (studentCourse.Course.FinalExamDate != null)
            {
                if (!studentCourse.FinalExamInscripted)
                {
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal1",
studentCourse.Course.Code),
                                String.Format("Examen Final de {0}", courseTitle),
                                String.Format("Recuerde inscribirse al Examen Final de
{0}", courseTitle),
                                studentCourse.Course.FinalExamDate,
                                96);
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal2",
studentCourse.Course.Code),
                                String.Format("Examen Final de {0}", courseTitle),
                                String.Format("Recuerde inscribirse al Examen Final de
{0}", courseTitle),
                                studentCourse.Course.FinalExamDate,
                                84);
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal3",
studentCourse.Course.Code),
                                String.Format("Examen Final de {0}", courseTitle),
                                String.Format("Recuerde inscribirse al Examen Final de
{0}", courseTitle),
                                studentCourse.Course.FinalExamDate,
                                73);
                    AddRemainderForCourseExam(String.Format("{0}ExamFinal",
studentCourse.Course.Code),
                                String.Format("Examen Final de {0}", courseTitle),
                                String.Format("En d�a de ma�ana se dicta el Examen Final
de  {0}", courseTitle),
                                studentCourse.Course.FinalExamDate,
                                24);
                }
                else
                {
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal1",
courseTitle));
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal2",
courseTitle));
                    _scheduleActionService.Remove(String.Format("{0}ExamFinal3",
courseTitle));
```

```
                _scheduleActionService.Remove(String.Format("{0}ExamFinal",
courseTitle));
            }


        }

        return Observable.Return(new TaskSummary
                                 {
                                     Result = TaskResult.Success
                                 });
    }
    catch (Exception)
    {
        return Observable.Return(new TaskSummary
                                 {
                                     Result = TaskResult.Success
                                 });
    }
}

    private void AddRemainderForCourseExam(string remainderName, string title, string
content, DateTime? examDate, int hoursEarlyToRemaind)
    {
        var firstPartialReminder = remainderName;
        if (examDate == null) return;
        var scheduledAction = _scheduleActionService.Find(firstPartialReminder);
        var beginTime = ((DateTime)examDate).AddHours(-hoursEarlyToRemaind);
        var reminder = new Reminder(firstPartialReminder)
            {
                BeginTime = beginTime,
                Title = title,
                Content = content
            };
        if (scheduledAction != null)
        {
            if (!scheduledAction.BeginTime.Equals(beginTime))
            {
                _scheduleActionService.Remove(firstPartialReminder);
                _scheduleActionService.Add(reminder);
            }
        }
        else
        {
            _scheduleActionService.Add(reminder);
        }
    }
```

```
    }
}
```

SyncronizationService.cs

```
using System;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services
{
    public class SyncronizationService : ISyncronizationService
    {
        private readonly ICloudService _cloudService;
        private readonly IStudentInformationManager _studentInformationManager;
#if DEBUG
        private readonly Uri _baseUri = new Uri("http://127.0.0.1:8081/Service1.svc/");
#else
        private readonly Uri _baseUri = new Uri("http://157.56.176.20:8081/Service1.svc/");
#endif
        public SyncronizationService(ICloudService cloudService,
IStudentInformationManager studentInformationManager)
        {
            _cloudService = cloudService;
            _studentInformationManager = studentInformationManager;
        }

        public IObservable<TaskSummary> SyncCourses(string careerCode, string
studentCode, string password)
        {
            var resuls = _cloudService.GetAllCourses(_baseUri, careerCode, studentCode,
password)
                .Select(student =>
                    {
                        var message =
_studentInformationManager.ProcessStudentInformation(student);
                        return new TaskSummary
                                {
                                    Result = TaskResult.Success
                                };
                    })
                .Catch((Exception exception) =>
                        {
                            throw exception;
                        });

            return resuls;
```

```
        }

        public IObservable<StudentCourse> SyncCourse(int code, string studentCode, string
    careerCode)
        {
            var result = _cloudService.GetCourse(_baseUri, code, studentCode, careerCode)
                                      .Select(course =>
                                         {

    _studentInformationManager.ProcessStudentCourseInformation(course);
                                             return course;
                                         })
                                      .Catch((Exception exception) =>
                                      {
                                         throw exception;
                                      });
            return result;
        }

        public IObservable<TaskSummary> RegisterNotificationChannelForStudent(string
    studentCode, string careerCode, string uri)
        {
            return _cloudService.RegisterNotificationChannelForStudent(_baseUri,
    studentCode, careerCode, uri).Select(summary => summary);
        }


        public IObservable<TaskSummary> CheckStudentForFinalExamOfCourse(string
    studentCode, string courseCode)
        {
            return _cloudService.CheckStudentForFinalExamOfCourse(_baseUri, studentCode,
    courseCode).Select(summary => summary);
        }
    }
}
```

Interfaces

ICloudService.cs

```
using System;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ICloudService
    {
```

```
        IObservable<Student> GetAllCourses(Uri baseUri, string careerCode, string
    studentCode, string password);
        IObservable<StudentCourse> GetCourse(Uri baseUri, int code, string studentCode,
    string careerCode);
        IObservable<TaskSummary> RegisterNotificationChannelForStudent(Uri baseUri,
    string studentCode, string careerCode, string uri);
        IObservable<TaskSummary> CheckStudentForFinalExamOfCourse(Uri _baseUri,
    string studentCode, string courseCode);
    }
}
```

IScheduleActionClient.cs

```
namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionClient
    {
        void AddPeriodicTask();
        void RemovePeriodicTask();
    }
}
```

IScheduleActionService.cs

```
using System;
using Microsoft.Phone.Scheduler;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IScheduleActionService
    {
        ScheduledAction Find(string taskName);
        void Add(ScheduledAction action);
        void Remove(string taskName);
        void LaunchForTest(string actionName, TimeSpan delay);
    }
}
```

IStudentInformationManager.cs

```
using System;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface IStudentInformationManager
    {
        string ProcessStudentInformation(Student student);
        void ProcessStudentCourseInformation(StudentCourse studentCourse);
        IObservable<TaskSummary> AddCourseExamsDatesReminders(StudentCourse
    studentCourse);
```

```
    }
}
```

ISyncronizationService.cs

```
using System;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Interfaces
{
    public interface ISyncronizationService
    {
        IObservable<TaskSummary> SyncCourses(string careerCode, string studentCode,
string password);
        IObservable<StudentCourse> SyncCourse(int code, string studentCode, string
careerCode);
        IObservable<TaskSummary> RegisterNotificationChannelForStudent(string
studentCode, string careerCode, string uri);
        IObservable<TaskSummary> CheckStudentForFinalExamOfCourse(string
studentCode, string courseCode);
    }
}
```

Mocks

MockCloudService.cs

```
using Microsoft.Phone.Reactive;
using System;
using System.Collections.Generic;
using TesinaMobileCloud.Data.DTO;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Phone.Services;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockCloudService : ICloudService
    {
        public IObservable<Student> GetAllCourses(Uri baseUri)
        {
            return Observable.ToObservable(new List<Student>{new Student
                {
                    Courses = new List<StudentCourse>
                        {
                            new StudentCourse
                            {
                                Course = new Course
                                {
```

```
                                Code = 1,
                                Title = "Programaci�n I",
                                Professor = "Ing. Carlos Rodrigo",
                                Description = "Programacion I introduce los conceptos basicos
de la programacion, "+
                                "comenzado con el paradigma estructurado y luego
introduciendo el paradigma de objetos. "+
                                "El lenguaje de programacion a utilizar es Java.",
                                TotalHoursOfClasses = 60,
                                Scheduled = "Miercoles 17Hs",
                                FinalExamDate = new DateTime(2013,02,04,00,05,00),
                                PartialExamDate = new DateTime(2013,02,04,00,08,00),
                                PracticWorkPresentationDate = new
DateTime(2013,02,04,00,07,00),
                                RecuperationExameDate = new DateTime(2013,02,04,00,06,00)
                            },
                            FinalExamResult = 10,
                            PartialExamResult = 9,
                            PracticWorkResult = 9,
                            RecuperationExamResult = 0,
                            TotalHoursAssisted = 40
                        }
                    }
            }));
        }

        public IObservable<StudentCourse> GetCourse(Uri baseUri, int code, string
studentCode, string careerCode)
        {
            throw new NotImplementedException();
        }

        public IObservable<TaskSummary> RegisterNotificationChannelForStudent(Uri
baseUri, string studentCode, string careerCode, string uri)
        {
            throw new NotImplementedException();
        }


        public IObservable<TaskSummary> CheckStudentForFinalExamOfCourse(Uri
_baseUri, string studentCode, string courseCode)
        {
            throw new NotImplementedException();
        }
    }
}
```

MockScheduleActionService.cs

```
using System;
using Microsoft.Phone.Scheduler;
using TesinaMobileCloud.Phone.Services.Interfaces;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockScheduleActionService : IScheduleActionService
    {
        private PeriodicTask periodic;
        public ScheduledAction Find(string taskName)
        {
            return periodic;
        }

        public void Add(ScheduledAction action)
        {
            periodic = new PeriodicTask("MockAdd");
        }

        public void Remove(string taskName)
        {
            periodic = null;
        }

        public void LaunchForTest(string actionName, TimeSpan delay)
        {
            //throw new NotImplementedException();
        }
    }
}
```

MockSyncronizationService.cs

```
using System;
using Microsoft.Phone.Reactive;
using TesinaMobileCloud.Phone.Services.Interfaces;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Mocks
{
    public class MockSyncronizationService : ISyncronizationService
    {
        public IObservable<TaskSummary> SyncCourses()
        {
            return Observable.Return(new TaskSummary
                                {
                                    Result = TaskResult.Success
```

```
                                    });
        }

        public IObservable<StudentCourse> SyncCourse(int code, string studentCode, string
    careerCode)
        {
            throw new NotImplementedException();
        }

        public IObservable<TaskSummary> RegisterNotificationChannelForStudent(string
    studentCode, string careerCode, string uri)
        {
            throw new NotImplementedException();
        }


        public IObservable<TaskSummary> CheckStudentForFinalExamOfCourse(string
    StudentCode, string CourseCode)
        {
            throw new NotImplementedException();
        }
    }
}
```

TesinaMobileCloud.Phone.Services.Test

CloudServiceFixture.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;
using TesinaMobileCloud.Data.DTO;

namespace TesinaMobileCloud.Phone.Services.Test
{
    [TestClass]
    public class CloudServiceFixture
    {
        [TestMethod]
        public void GetAllCourses()
        {
            var sut = new MockCloudService();
            var result = sut.GetAllCourses(new Uri("http://127.0.0.1"));
            Assert.IsNotNull(result);
        }
    }
}
```

ScheduleActionsServiceFixture.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;

namespace TesinaMobileCloud.Phone.Services.Test
{
   [TestClass]
   public class ScheduleActionsServiceFixture
   {
      public Interfaces.IScheduleActionService mockScheduleService = new
MockScheduleActionService();

      [TestMethod]
      public void RemoveAgent()
      {
         var sut = new ScheduleActionClient(mockScheduleService);

         sut.RemovePeriodicTask();

         Assert.IsNull(mockScheduleService.Find("MockRemove"));

      }

      [TestMethod]
      public void AddAgent()
      {
         var sut = new ScheduleActionClient(mockScheduleService);

         sut.AddPeriodicTask();

         Assert.IsNotNull(mockScheduleService.Find("MockAdd"));
      }

   }
}
```

SyncronizationServiceFixture.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TesinaMobileCloud.Phone.Services.Mocks;

namespace TesinaMobileCloud.Phone.Services.Test
{
   [TestClass]
   public class SyncronizationServiceFixture
   {
      [TestMethod]
```

```
        public void SyncCourses()
        {
            var sut = new MockSyncronizationService();

            var result = sut.SyncCourses();

            Assert.IsNotNull(result);
        }

        //[TestMethod]
        //public void SyncCourse()
        //{
        //    var code = 1;
        //    var sut = new MockSyncronizationService();

        //    var result = sut.SyncCourse(code);

        //    Assert.IsNotNull(result);
        //}
    }
}
```

TesinaMobileCloud.Phone.Shared

Class1.cs

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using GalaSoft.MvvmLight.Ioc;

namespace TesinaMobileCloud.Phone.Shared
{
    public class CustomIocContainer : SimpleIoc
    {
        static CustomIocContainer()
        {

        }
    }
```

TesinaMobileCloud.PushNotificationSenderRole

PushNotificationSenderRole.cs

```
using System;
using System.Diagnostics;
using System.Net;
using System.Text;
using System.Threading;
using Microsoft.WindowsAzure.ServiceRuntime;
using Ninject;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Shared;

namespace TesinaMobileCloud.PushNotificationSenderRole
{
    public class PushNotificationSenderRole : RoleEntryPoint
    {
        private IKernel _container;
        private IQueue<QueueNotificationMessage> _queue;
        //private IQueue<string> _errors;
        public override void Run()
        {
            _queue = _container.Get<IQueue<QueueNotificationMessage>>();
            //_errors = _container.Get<IQueue<string>>();

            while (true)
            {
                Thread.Sleep(10000);
                if (!_queue.HasMessage) continue;
                var notificationMessage = _queue.GetMessage();

                try
                {
                    SendNotification(notificationMessage);
                }
                catch (Exception ex)
                {
                    _queue.AddMessage(notificationMessage);
                }
            }
        }

        private static void SendNotification(QueueNotificationMessage notificationMessage)
        {
            var sendNotificationRequest =
WebRequest.Create(notificationMessage.Destinatary);
```

```
            var notificationMessageInBytes = new
UTF8Encoding().GetBytes(notificationMessage.Payload);

            sendNotificationRequest.Method = WebRequestMethods.Http.Post;
            sendNotificationRequest.ContentType = "text/xml";
            sendNotificationRequest.ContentLength = notificationMessageInBytes.Length;
            sendNotificationRequest.Headers["X-MessageID"] = Guid.NewGuid().ToString();

            sendNotificationRequest.Headers.Add("X-WindowsPhone-Target",
notificationMessage.Type);
            sendNotificationRequest.Headers.Add("X-NotificationClass",
notificationMessage.Priority);


            using (var requestStream = sendNotificationRequest.GetRequestStream())
            {
                requestStream.Write(notificationMessageInBytes, 0,
notificationMessageInBytes.Length);
            }

            // Send the notification and get the response.
            var response = sendNotificationRequest.GetResponse();
            var notificationStatus = response.Headers["X-NotificationStatus"];
            var notificationChannelStatus = response.Headers["X-SubscriptionStatus"];
            var deviceConnectionStatus = response.Headers["X-DeviceConnectionStatus"];
            //if(!notificationStatus.Equals("Connected") ||
!deviceConnectionStatus.Equals("Received"))
        }

        public override bool OnStart()
        {
            // Set the maximum number of concurrent connections
            ServicePointManager.DefaultConnectionLimit = 12;

            // For information on handling configuration changes
            // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.
            _container = new StandardKernel(new ServiceModule());


            return base.OnStart();
        }
    }
}
```

TesinaMobileCloud.Shared

ServiceModule.cs

```
using Microsoft.WindowsAzure;
using Ninject.Modules;
using TesinaMobileCloud.Data;
using TesinaMobileCloud.Data.Model;
using TesinaMobileCloud.Data.Queue;
using TesinaMobileCloud.Data.Repository;
using TesinaMobileCloud.Data.Repository.Interfaces;
using TesinaMobileCloud.Data.Table;

namespace TesinaMobileCloud.Shared
{
   public class ServiceModule : NinjectModule
   {
      public override void Load()
      {
         Bind<CloudStorageAccount>().ToMethod(context =>
CloudStorageConfiguration.GetCloudAccount("DataConnectionString"));

         Bind<ITable<Course>>().To<AzureTable<Course>>();
         Bind<ITable<Career>>().To<AzureTable<Career>>();

Bind<ITable<CareerCourseRelationship>>().To<AzureTable<CareerCourseRelationship>>(
);
         Bind<ITable<Student>>().To<AzureTable<Student>>();

Bind<ITable<StudentCourseRelationship>>().To<AzureTable<StudentCourseRelationship>
>();


Bind<IQueue<QueueNotificationMessage>>().To<NotificationQueue<QueueNotificationMe
ssage>>();
         Bind<IQueue<string>>().To<PushNotificationErrorsQueue<string>>();

         Bind<ICareerRepository>().To<CareerRepository>();
         Bind<ICourseRepository>().To<CourseRepository>();

Bind<ICourseCareerRelationshipRepository>().To<CareerCourseRelationshipRepository>()
;
         Bind<IStudentRepository>().To<StudentRepository>();
         Bind<IStudentCourseRepository>().To<StudentCourseRepository>();
      }
   }
}
```