UNIVERSIDAD DE BELGRANO

Las tesinas de Belgrano

Facultad de Ingeniería y Tecnología Informática
Carrera de Ingeniería en Informática

Desarrollo de Aplicaciones Basadas en XML Web Services para Dispositivos Móviles con Microsoft .NET Compact Framework

Nº 155 **Pablo Andrés Bianco**

Tutor: Carlos G. Said

Departamento de Investigación
Abril 2005

Agradecimientos

A Bruno y Silvia Bianco, mis padres, quienes me han *dado todo con amor y desinteresadamente.*

Agradezco mucho a todas las personas que me han formado en mi camino universitario como ingeniero en informática, a la **Universidad de Belgrano**, institución donde me formé y que además me permitió haber conocido nuevos horizontes en otros países donde pude estudiar y crecer, a Carlos Said, mi tutor, quien me ha conducido en esta tesina con entusiasmo, suma dedicación y me ha tratado constantemente como a un profesional aunque no lo fuera, al personal de Microsoft de Argentina quienes me han brindado amablemente material académico. Y nuevamente, a mis padres y familia.

Indice

Introducción

Nos encontramos actualmente en lo que algunos visionarios de la industria informática denominan la «década digital», caracterizada por una importante tendencia hacia las formas digitales. Lo vemos a diario en las operaciones bancarias y comerciales, en las fotografías, en las nuevas formas de audio y video, y en los documentos que enviamos y recibimos por Internet. El volumen de información digitalizada ha crecido considerablemente y ya ha pasado a ser de uso prácticamente cotidiano.

El avance en las tecnologías de comunicaciones ha sido sin duda una de las causas más importantes que ha incrementado indirectamente el manejo de la información digital, y ha sentado las bases para muchas áreas relacionadas. Con Internet y redes privadas disponibles a un costo relativamente bajo y con un ancho de banda adecuado ha sido posible intercambiar todo de tipo de información, precisamente, digital.

Paralelo al avance de las comunicaciones, las industrias electrónicas y de informática se han desarrollado continuamente. Las organizaciones y personas individuales sacan provecho de los avances. Ya no solo es posible contar con sistemas de información en una empresa sino que se puede trascender el ámbito físico y llevar los sistemas prácticamente a cualquier lugar donde las comunicaciones estén disponibles. Es aquí donde los dispositivos móviles adquieren un rol protagónico. Estos dispositivos pueden contener aplicaciones personales, y empresariales, como una extensión de los sistemas de la organización. Así, las empresas encuentran nuevas oportunidades de negocio, la posibilidad de acceder a nuevos mercados y alternativas para incrementar la ventaja competitiva.

En esta tesina se trata el desarrollo de aplicaciones para dispositivos móviles de la mano de una tecnología concreta, el Compact Framework. Al principio se exponen las características de estos equipos, cuáles son las perspectivas actuales y futuras para su uso, y sus limitaciones principales. Posteriormente se introduce el marco de trabajo propuesto para el desarrollo de aplicaciones móviles, las cuales requieren algunas consideraciones especiales de diseño e implementación. Se dan pautas para lograr un desarrollo eficiente en bases a las restricciones de hardware y software actuales.

Como nexo de comunicación entre una aplicación móvil y los sistemas organizacionales se propone el uso de XML Web Services, que suponen una forma relativamente simple de comunicar sistemas heterogéneos de manera transparente, basándose en la infraestructura ya existente de Internet.

Por último, se incluye el desarrollo de un escenario en el cual una aplicación .NET móvil accede a servicios Web ofrecidos por una universidad. Así, los alumnos de la universidad pueden acceder a la base de datos para realizar consultas sobre cursos disponibles e inscribirse a los mismos, realizando esto desde un dispositivo móvil.

1.1 Conceptualización de los dispositivos móviles

Según el Diccionario de la Real Academia Española, un **dispositivo** es cualquier mecanismo o artificio dispuesto para producir una acción prevista. Podemos extender este término para hablar de «dispositivos portátiles», que en el sentido amplio señala a dispositivos que una persona puede llevar consigo para su utilización y que ofrecen alguna funcionalidad o prestan algún servicio. Definidos así de manera inespecífica, los teléfonos celulares, discman, calculadoras y todos aquellos que cumplan esta definición general son dispositivos portátiles.

Distinto es el término **dispositivo móvil.** En [HK02.0] se los define como aparatos electrónicos que sirven para la comunicación, procesamiento e intercambio de datos y pueden ser llevados por sus usuarios para enviar, recibir o compartir datos con otros dispositivos. [KO03.0] comparte con la fuente anterior una definición por extensión de los mismos, donde se incluyen:

1. teléfonos celulares
2. computadoras portátiles (laptops, notebooks)
3. PDAs[1]
4. híbridos de los anteriores

En el ámbito de esta tesina vamos a darle una serie de características propias de acuerdo a los requerimientos del marco de trabajo que posteriormente se desarrollará. De aquí en adelante, cuando se mencionen los dispositivos móviles, el lector debe asociar a ellos las siguientes propiedades:

- son dispositivos electrónicos computacionales
- se pueden trasladar fácilmente por su forma y tamaño adecuado
- inalámbricos
- tienen autonomía eléctrica, es decir, pueden ser operados con independencia de la red eléctrica por un lapso de tiempo determinado

1. En inglés, Personal Digital Assistant que significa Asistente Personal Digital

- son capaces de conectarse a redes de datos
- actualmente tienen restricciones de hardware y software suficientes para requerir un tratamiento especial
- poseen una CPU[2] con capacidad adecuada de procesamiento
- tienen memoria RAM[3] para el almacenamiento y ejecución de programas
- poseen formas de almacenamiento permanente

Las últimas tres características le confieren la cualidad de hallarse encuadrados en una clase denominada **dispositivos inteligentes**[4] [MS01.1].

Dispositivos de Consideración Especial

Es importante indicar que existen equipos electrónicos de avanzadas características que no son objeto de estudio de esta tesina, y sobre los cuales actualmente no pueden desarrollarse aplicaciones con el marco de trabajo que nos ocupa. Este es el caso, por ejemplo, de **aparatos de telefonía celular** que permiten acceder a servicios de mensajería y hasta tienen micronavegadores de Internet, pero no cuentan con la capacidad de ejecutar aplicaciones personalizadas.

Cabe indicar que las computadoras portátiles **notebooks o laptops** no serán consideradas en la categoría de dispositivos móviles. La razón es que no poseen las restricciones fundamentales[5] para requerir un tratamiento especial a la hora de desarrollar aplicaciones. Aunque poseen una arquitectura reducida en tamaño, ofrecen las mismas facilidades que una PC de escritorio. Un notebook actual puede funcionar con el mismo sistema operativo que una computadora de escritorio y puede ejecutar los mismos entornos de desarrollo de aplicaciones, tienen teclado completo, acceso a las mismas unidades periféricas, display de gran tamaño, etc. La cuestión específica que las excluye de esta categoría es el diseño de las aplicaciones, que sigue las mismas reglas que se emplean en una PC de escritorio en cuanto a sus interfaces de usuario y uso de los recursos. No se requiere el Compact Framework[6] y de hecho no está disponible para PC.

Las tablet PCs en cambio serán consideradas como dispositivos móviles. Esto puede parecer paradójico según lo indicado anteriormente para los notebooks, pues las tablet PCs tampoco tienen las restricciones fundamentales mencionadas. La diferencia radica en que éstas han sido diseñadas y optimizadas para su uso móvil. Tienen características hardware que las hace muy adecuadas para integrarlas en las actividades diarias de las personas: son livianas, ofrecen características extendidas para la entrada/salida de datos (pueden usarse como un práctico tomador de apuntes por medio de la pantalla sensible al tacto, como libro de lectura) y pueden ser utilizadas como PC portátil. La idea detrás de esto es que son mucho más prácticas que un notebook para el contexto móvil.

Habiendo realizado las aclaraciones necesarias, algunos ejemplos estándar de lo que consideramos dispositivos móviles son:

PDA **XDA** **SmartPhone** **Tablet PC**

(PDA + teléfono)

1.2 Perspectiva futura del uso de dispositivos móviles

Al momento de fundar su empresa en 1975, Bill Gates tuvo una simple pero muy acertada visión [MS01.0]: «cada hogar tendrá en unos años una computadora personal disponible», lo cual actualmente parece ser evidente pero en aquel tiempo no lo era tanto debido a los altos costos del hardware y la escasa disponibilidad de software para ser adquirido masivamente. Michael Dertouzos[7], unos años antes de la aparición al

2. En inglés, Central Process Unit que indica Unidad Central de Proceso
3. En inglés, Random Access Memory que significa Memoria de Acceso Aleatorio.
4. En inglés, «Smart Devices».
5. Tales restricciones se exponen en la sección 4 de este capítulo.
6. El Compact Framework de Microsoft es un marco de trabajo reducido diseñado especialmente para dispositivos móviles de capacidades limitadas de hardware y software. Será tratado con más detalles en el capítulo 2.
7. Director del Laboratorio de Ciencias de la Computación del MIT, Massachusetts Institute of Technology.

mercado de las primeras computadoras personales le indicó a un periodista de la revista *People* que los costos de las mismas serian accesibles y que en el plazo de una década uno de cada tres hogares tendría una PC. Añadió además que, debido al avance de la tecnología, las redes de ordenadores también iban a proliferar.

En el año 1999 tras haber comprobado en cierta medida su visión original, Gates reformuló su perspectiva de acuerdo a los avances del momento e indicó que: «la gente se vería beneficiada por software poderoso disponible en cualquier lugar, en cualquier momento y en cualquier dispositivo».

De la reformulación de la visión de Gates surgen al menos tres preguntas esenciales. La primera es ¿qué hace posible la disponibilidad del software en cualquier lugar?, ¿qué hace posible la disponibilidad del software en cualquier momento? y ¿qué hace posible la disponibilidad del software en cualquier dispositivo?

Para lograr estas premisas se requiere una ***infraestructura*** que lo soporte, no obstante posibles respuestas simplificadas podrían ser: conectividad y acceso a redes de alta disponibilidad para la primera y segunda pregunta, estándares y marcos de trabajo independientes de plataforma[8] para la tercera pregunta.

La tercera pregunta expone, de alguna forma, la problemática general a la que se dirige esta tesina. La expresión genérica «cualquier dispositivo» incluye a los dispositivos móviles, que han sido referenciados directa e indirectamente por varios expertos y visionarios del ambiente tecnológico, como Bill Gates y Michael Dertouzos.

Dertouzos en [DE97] imagina algunas facilidades que estarán disponibles para la gente, basadas en su visión sociotecnológica. Por ejemplo, habla de *automóviles informados,* conectados remotamente a los servidores de la compañía, con sistemas de posicionamiento global que permiten indicar latitud y longitud exacta, mapas en línea e información de navegación de rutas, con capacidades de traslación autónoma, y varias otras características que hoy en día no son habituales pero que ya se han comenzado a ver.

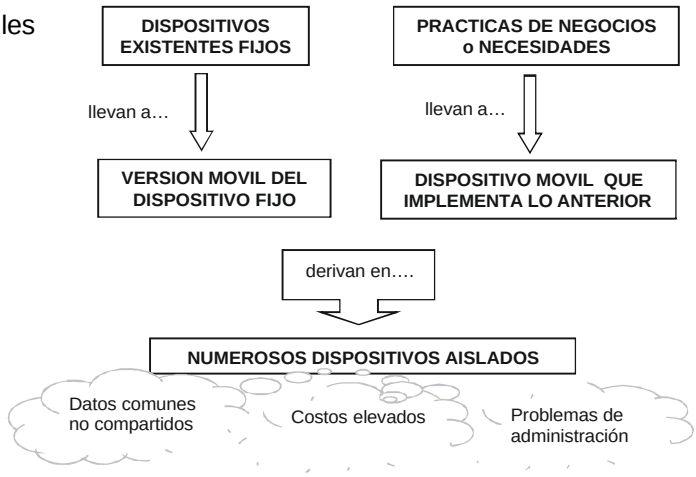
Gates fundamenta su nueva visión señalando dos principales décadas de la computación actual [MS03.1]:

- Del 1990 al 1999, denominada «Década de la PC», caracterizada por la presencia de sistemas operativos monoprogramados[9] al inicio, dando paso a multiprogramados con poderosas interfaces gráficas y la presencia de protocolos de Internet. La difusión de estándares dio «poder a las masas», esto es, mediante la adopción de estándares reconocidos comenzó a hacerse posible la integración de tecnologías y comunicaciones a gran escala.
- Del 2000 al 2009, denominada «Década Digital», caracterizada por la proliferación de los dispositivos móviles, la alta integración de tecnologías y donde las fotografías, audio y video tienden hacia la forma digital. El desarrollo de la tecnología móvil ha sido lento porque los avances aún están ocurriendo: ampliación de capacidades de almacenamiento, incremento en la velocidad de procesadores, expansión de redes inalámbricas. Remarca la integración de distintas tecnologías móviles.

Esta década no marca la sustitución de PCs por dispositivos móviles ni viceversa, pues considera un nuevo entorno computacional formado por una amplia gama de dispositivos desde largas pantallas planas hasta los más pequeños dispositivos móviles, PDAs, Smartphones, etc. «El poder está en la **integración»**, pone el ejemplo de una persona que adquirió un boleto de avión y ante un eventual cambio de vuelo, el usuario recibe un alerta en su dispositivo móvil. «Queda mucho por hacer aún»- agregó.

A continuación se expone una caracterización de las tendencias de dispositivos tecnológicos en general según he analizado, seguida por una explicación:

Figura 1.1 Tendencias de los dispositivos móviles



8. Tanto de hardware como de software

9. En la terminología de los sistemas operativos, se refiere a aquellos que son capaces de ejecutar un único proceso completo por vez, sin posibilidad de cambios de contexto. Debe finalizarse la aplicación actual para dar paso a una nueva.

Resulta evidente que ante esta perspectiva futura en donde el hombre tiene la necesidad de acceso permanente e integral a la información, los dispositivos móviles tendrán un papel protagónico por la practicidad y flexibilidad que ofrecen.

Por un lado ha habido una tendencia a desarrollar versiones portátiles de muchos dispositivos fijos que se usan habitualmente. Este es el caso de la telefonía clásica que derivó en celulares, el caso computadoras de escritorio que derivó a PCs portátiles, de software de organización de tareas y contactos que derivó en simples agendas electrónicas hasta las más avanzadas que tenemos actualmente, etc.

Por otro lado ha habido una tendencia a desarrollar dispositivos portátiles para implementar prácticas de negocio/necesidades en cualquier lugar. Este sería el caso de equipos de recolección de datos portátiles utilizados por empresas de servicios para tomar mediciones y volcarlas en aplicaciones centrales, el caso de los equipos utilizados por vendedores de campo con los cuales levantan sus pedidos en la calle, de instrumentos específicos de ubicación geográfica como los dispositivos de GPS manuables, etc.

Ante una nueva variedad de dispositivos móviles donde muchos comparten características comunes (datos compartidos) pero funcionalidades específicas diferentes, surgen algunos problemas que requieren solución: es claro ver que una persona no puede desenvolverse adecuadamente con múltiples dispositivos portátiles diferentes, pues resulta impráctico y costoso. Además, si varios equipos comparten datos comunes se generan problemas de administración: inconsistencias, pérdidas de datos, dificultad de actualización y de uso.

Se plantea la necesidad de un nuevo diseño:

1. unificación del hardware con funcionalidad
2. sincronización de los datos
3. procesamiento distribuido de aplicaciones

La tendencia de **unificación del hardware con funcionalidad** tiene que ver con unir funciones ofrecidas por varios dispositivos en un único equipo. Un ejemplo complejo y real a la vez es: un ejecutivo que se traslada por distintos sitios geográficos necesita estar comunicado, tener acceso a documentos y planillas de cálculo. Se vería beneficiado con el uso de un equipo portátil pequeño con capacidad propia procesamiento y almacenamiento que permita ejecutar aplicaciones de negocios, personales, de entretenimiento y comunicaciones (telefonía tradicional o IP, internet), que provea una interfaz para la conexión y sincronización con otros dispositivos o sistemas centrales residentes en hosts.

La **sincronización de datos** consiste en homogeneizar los contenidos de los repositorios de datos de los dispositivos móviles con un host que posee el usuario, u otros dispositivos. Se trata de llevar las bases de datos a un estado de convergencia para que haya disponibilidad de datos, facilidad de actualización y evitar pérdidas e inconsistencias.

En la industria informática actual tenemos muchas empresas que desarrollan productos y soluciones móviles. Para el caso particular de los asistentes personales digitales (PDAs), hay más de un sistema operativo disponible y cada proveedor equipa al PDA con distintos componentes de hardware y de software. ¿Cómo se puede hacer para que se comuniquen entre sí dispositivos y sistemas no homogéneos? Para ello la industria[10] ha establecido y adoptado estándares que regulan la forma de comunicación y presentación de los datos. Las empresas de la industria compiten por ofrecer herramientas basadas en estos estándares ya aceptados.

El **procesamiento distribuido de aplicaciones** consiste en la comunicación abierta, independiente de características de implementación específica, entre dispositivos locales o distantes, que pueden compartir aplicaciones, datos u otros recursos.

Dertouzos [DE97] indica que a mediados del año 1996, el Web Consortium ya contaba con 150 organizaciones en calidad de miembros, entre las cuales se destacaban AT&T, Microsoft, Sony, Netscape y Sun, entre otras. Para ese momento sus miembros ya comprendían la necesidad de un consenso, de la adopción de estándares como elemento preservador de la integridad de la Web compuesta por millones de máquinas del mundo entero y de ganar dinero con *software y servicios ampliamente compartidos basados en la Web*. Esta última frase da lugar a algunas interrogantes: ¿software y servicios ampliamente compartidos a través de dispositivos móviles entre otros? ¿es aquí donde surge la idea de los Web Services? A lo largo de esta tesina mostraré el desarrollo de aplicaciones móviles basadas en XML Web Services, como posible respuesta a estas preguntas.

¿Cuál es la perspectiva futura para estos equipos y aplicaciones según algunas de las consultoras más relevantes?

En [IC01] y [GA01] se muestran estudios de mercado hecho en el año 2001 proyectados para los años 2004 y 2005.

10. En particular, el Consorcio de Internet www.w3c.org

Gartner Group señala que para el año 2005 se usarán a lo largo del mundo alrededor de mil millones de dispositivos móviles en general.

Los investigadores de Gartner Research perciben tres razones para la inversión en aplicaciones móviles:

1. Oportunidad de *inversión*. La organización puede realizar inversiones cuando las tecnologías móviles ofrecen oportunidades potenciales de negocio que generen nuevos ingresos. Ejemplo: portal de compras desde dispositivos móviles.
2. Respuesta ante *amenaza*. La implementación de aplicaciones móviles en algunas áreas de negocio tal vez no genere nuevos clientes. No obstante, el costo de la no implementación puede ser pérdidas de clientes actuales.
3. *Ahorro* de costos. Por ejemplo: reemplazo de procesos basados en papel.

Este grupo de investigación espera que, en diez años a partir del 2001, los dispositivos móviles tengan un rol principal en el trabajo y en la sociedad. Se los visualiza en un futuro con las siguientes características: siempre encendidos (listos para usar), inalámbricos y adecuados para que los usuarios los lleven siempre consigo. Se plantea, además, que los dispositivos móviles habilitan la llamada «Supranet», una Internet ampliada que provee integración y comunicación entre dispositivos, integración de los procesos de negocios e interfaces de comunicación entre los dispositivos y las personas. El grupo indica que hay elementos clave habilitadores de la Supranet:

- Redes inalámbricas
- Servicios de localización geográfica de dispositivos
- Nuevas formas (o formas mejoradas) de tecnologías de entrada/salida que permitan mejorar la usabilidad de los dispositivos
- Proliferación y mejora de los dispositivos computacionales personales

A través de un modelo de estimación de tendencias en el mercado denominado *Technology Radar Screen* (pantalla de radar de tecnología), Gartner realiza una «suposición» para un planeamiento estratégico. La suposición indica que, para el 2005, las tecnologías que tendrán crecimiento rápido son: XML, tecnologías Web inalámbricas, procesamiento del lenguaje natural, Bluetooth, esquemas de pagos electrónicos, autorización digital, etc.; con una probabilidad del 70%. Entre las tecnologías mencionadas, podemos ver que varias de ellas están relacionadas estrechamente al tema central de esta tesina: XML (usado como lenguajes para los servicios Web), tecnologías Web inalámbricas (como parte de la infraestructura móvil), Bluetooth (como estándar para comunicaciones entre dispositivos), etc.

McKinsey & Co, por otra parte, indicó que para este año (2004) las empresas podrían ahorrar alrededor de 80 mil millones de dólares al año con el uso de aplicaciones móviles por empleados, clientes y partners.

1.3 Perspectiva actual del uso de dispositivos móviles

¿Por qué usar dispositivos móviles en las empresas?

Se plantean dos motivos principales [MS01.1], de los cuales se derivan muchas prestaciones.

El primer motivo es el **costo del hardware** que continuamente decrece en relación a sus prestaciones. Los principales proveedores del mercado compiten ofreciendo nuevas características y costos menores que induce a las empresas a considerar inversiones para la adquisición de dispositivos para un gran número de empleados.

Los dispositivos son cada vez más pequeños, más poderosos en sus capacidades de procesamiento y almacenamiento, e incorporan nuevas tecnologías que flexibilizan las tareas de rutinarias de las empresas; por ejemplo, el empleo de tecnologías de comunicación inalámbricas.

El segundo motivo es una **mejorada arquitectura de software** disponible para estos dispositivos. Existen industrias de software que ofrecen marcos de trabajo que facilitan mucho el proceso de desarrollo de software.

No se plantea el uso de los dispositivos móviles como sustitutos de las computadoras personales de escritorio que corren sistemas organizaciones, sino como una extensión de los mismos.

La entidad que incorpora dispositivos móviles para su operación puede obtener una serie de ventajas relativas [MS01.0]:

- facilidad para la administración de información personal
- manipulación de documentos: informes, planillas, imágenes, etc.
- recolección de datos automática con mecanismos de validación
- reducción de costos de operación
- seguimiento de stock para vendedores que trabajan afuera de sus oficinas
- acceso remoto a información del cliente sobre su estado de cuenta
- incrementar la satisfacción del cliente ofreciendo un servicio más eficiente

· en el caso de entidades educativas, podría ser una herramienta integral para asistir al proceso enseñanza y aprendizaje.

Los puntos recientemente citados son sólo ejemplos. Las distintas organizaciones pueden emplear estas características básicas y adaptarlas a sus necesidades concretas.

En el mundo, millones de personas trabajan fuera de los entornos tradicionales de trabajo. Muchos emplean métodos costosos e ineficientes para la captura de datos necesaria para completar sus tareas.

Podría definirse un abanico de los tipos de usuarios que se consideran más propensos a necesitar soluciones móviles, como es el caso de ejecutivos, representantes de ventas, ingenieros, médicos, conductores de vehículos, técnicos, personal cuyo lugar de trabajo es temporal, etc.

Con costos relativamente bajos, más eficientes y con facilidades para el desarrollo de software, los dispositivos portátiles se pueden considerar adecuados para ser componentes de gran importancia en las aplicaciones organizacionales.

¿Qué capacidades operativas se requieren para hacer que las aplicaciones de negocios estén disponibles para los usuarios móviles?

[MS01.2] plantea tres capacidades fundamentales. A saber:

1. Infraestructura
2. Opciones para la conectividad flexible
3. Entorno de desarrollo familiar

1. Infraestructura

Para que las aplicaciones cumplan los objetivos de sus usuarios deben reunir una serie de características concretas. Deben ser:

· seguras: un dispositivo móvil está expuesto a riesgos de robo, pérdida, espionaje, etc.
· escalables: las aplicaciones debe acompañar el ritmo del crecimiento empresarial.
· administrables: cuando una compañía cuenta con un gran conjunto de dispositivos móviles, deben haberse definidos mecanismos de administración centralizada por medio de redes computacionales y software para su sincronización.

2. Opciones de Conectividad Flexible

Para aquellos usuarios que requieren algún tipo de conectividad, es muy importante que los dispositivos móviles sean compatibles con una variedad de opciones de conectividad por medio de tarjetas de hardware o puertos, de manera que el usuario tenga la posibilidad de implementar la arquitectura de comunicación más apropiada, sea Wi-Fi (en cualquiera de sus versiones), Bluetooth, tarjetas adaptadoras de red (NICs), módems, etc.

3. Entorno de desarrollo familiar

Si las herramientas disponibles para el desarrollo de aplicaciones móviles introducen una considerable complejidad adicional, la organización deberá abordar altos costos extras de capacitación y entrenamiento del personal de desarrollo en este nuevo entorno.

[MS01.2] indica que existen más de 5 millones de desarrolladores que emplean herramientas de uso generalizado tales como Visual Basic, Visual C++, ADO, COM, etc. Analizando esta cifra se desprende la idea que, existiendo alguna forma de desarrollar aplicaciones móviles con este tipo de herramientas, los costos de desarrollo no se incrementarían y algo aún más valioso, podría aprovecharse la *experiencia* adquirida que conduciría a desarrollos en menor tiempo y con costos viables.

¿Qué hace a los dispositivos y aplicaciones móviles aptos para brindar soluciones?

Douglas Dedo expone en [MS03.0] que, a medida que las organizaciones visualizan oportunidades de hallar soluciones con el uso de estos dispositivos, se requiere de un proceso formal que justifique la inversión requerida. No tiene el mismo atractivo hacer una inversión si los usuarios consideran que estos equipos son «anotadores modernos» en los que se puede chequear mail en caso de que una PC no funcione, que hacer una inversión si se identifican aplicaciones específicas de negocios que extienden las capacidades de los sistemas para cubrir áreas que otorguen ventajas.

Hay cuatro tendencias de la industria que les dan atractivos particulares:

1. Sistemas móviles como extensión de los sistemas empresariales
2. Convergencia a los dispositivos móviles
3. Mayor capacidad de almacenamiento
4. Mayor ancho de banda en conectividad inalámbrica

1. Sistemas móviles como extensión de los sistemas empresariales

Las empresas han estado haciendo que los datos operacionales sean accesibles por más usuarios. Se ve en los inicios de los primeros sistemas basados en mainframes, donde los datos comenzaban a ser accedidos desde PCs de escritorio. Desde entonces, el enfoque ampliamente aceptado ha sido la arquitectura cliente servidor. Actualmente se desea extender estos sistemas con aplicaciones basadas en Web, donde los datos pueden ser accedidos desde dispositivos más pequeños y portátiles ampliando así la infraestructura de la empresa.

2. Convergencia a los dispositivos móviles

Se plantea la idea ya expuesta anteriormente en la caracterización de los dispositivos tecnológicos en general, donde distintos dispositivos convergen en otros que combinan funcionalidades.

3. Mayor capacidad de almacenamiento

Los equipos móviles que soportan operaciones de negocios requieren almacenar una gran cantidad de datos. Las manufactureras de hardware han desarrollado una gama de medios de almacenamiento portátil de capacidades variadas, cuyos costos han decrecido con el tiempo.

4. Mayor ancho de banda en conectividad inalámbrica

Ya pueden observarse proveedores de conexiones a redes inalámbricas. El ancho de banda para conectividad local y de área amplia se ha venido incrementado en los últimos tiempos y los operadores inalámbricos han hecho énfasis en ampliar las áreas de cobertura y proveer servicios confiables.

La idea de los sistemas que implementan tecnologías móviles es en última instancia la de «sistemas sin fronteras». No solo es posible que las aplicaciones trasciendan el ámbito físico de la empresa (lo cual ya fue posible mediante redes computacionales) sino que además, su utilización no debe confinarse a espacios fijos. El usuario cuenta con la libertad de trasladarse y operar su sistema ampliado.

¿Cuáles son los beneficios ECONÓMICOS de la implementación de tecnologías móviles?

Douglas Dedo en [MS03.0] reconoce que hay «enfoques» predeterminados que son los que tienen mayor probabilidad de producir un rápido retorno de la inversión a la organización. Esto implica que se debe evaluar de antemano el tipo de proyecto que se quiere implantar. No se garantiza el éxito porque se trate de una solución móvil.

Enfoques:

Conversión de procesos basados en papel a aplicaciones basadas en formularios

La recolección de datos con papel es inherentemente lenta, inexacta y propensa a errores. La migración de procesos de recolección manual a procesos automatizados mediante formularios electrónicos en dispositivos móviles tiene la cualidad de acelerar la captura de los datos, facilitar el flujo y tener un control sobre las entradas. Los beneficios económicos se evidencian en:

- mayor número de transacciones posibles en el mismo tiempo que un proceso manual
- reducción de altos costos para la corrección de errores
- supresión del costo de ingreso de datos al sistema principal
- incremento en la satisfacción de los clientes debido a una atención más eficiente

Renovación de antiguos dispositivos portátiles

El mantenimiento y soporte necesario que requieren los antiguos equipos móviles, tales como lo que usan MS-DOS, tiene un costo creciente. El reemplazo de éstos por equipos actuales disminuye el costo total de propiedad de los mismos y los nuevos dispositivos permiten explotar las ventajas tecnológicas modernas que reducen los costos. Las variables en las que se aprecian reducciones son:

- costos en la actualización de hardware
- costos del desarrollo de software, debido a herramientas de generación automática de código, entornos familiares, etc.
- costos de las comunicaciones inalámbricas, pues lo dispositivos modernos pueden transmitir más datos en la misma unidad de tiempo

Aplicaciones basadas en Web

Hay organizaciones que necesitan que sus aplicaciones tengan un amplio alcance, accesibles para todo público posible. En estos casos, no es posible ser elitistas a la hora de determinar qué dispositivos soportarán las aplicaciones; deben ser accedidas desde múltiples dispositivos fabricados por las manufactureras existentes y en diversas plataformas de hardware y software. Para ello existen lenguajes para la representación e intercambio de datos, como XML, que combinados con soluciones de empresas particulares ofrecen un buen grado de abstracción respecto de los detalles de implementación específicos. Los beneficios económicos se observan en:

- los costos de desarrollo de sistemas cooperativos. Ejemplo: el proveedor de autopartes que suministra piezas de forma automática por medio de una interfaz de software con la automotriz
- los costos de desarrollo de ampliación de líneas de venta
- los costos de desarrollo y mantenimiento en general.

¿Qué partes clave intervienen en el desarrollo de soluciones móviles?

Se requiere el trabajo de diversos grupos industriales y organizaciones para lograr soluciones móviles integrales. [MS01.0] y [MS03.1] indican los siguientes sectores:

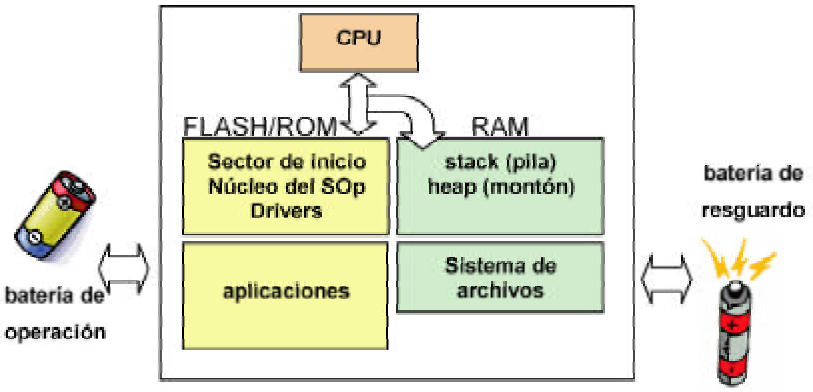
- manufactureras de dispositivos móviles
- manufactureras de dispositivos periféricos y de chips
- proveedores de servicios de red inalámbricos
- proveedores de soluciones móviles (proveen la experiencia)
- desarrolladores de software empresarial para dispositivos móviles

Considero oportuno agregar los siguientes sectores:

- proveedores de software y entornos de desarrollo para dispositivos móviles
- organismos de estandarización

1.4 La Arquitectura de un Dispositivo Móvil

Figura 1.2 Arquitectura de Pocket PC



Arquitectura de Hardware Pocket PC

La figura 1.2 muestra la arquitectura de un dispositivo móvil Pocket PC estándar. Existen tres componentes principales:

- la unidad central de proceso (CPU)
- una unidad de memoria de sólo lectura (que puede ser tecnología Flash)
- una unidad de memoria RAM

La **unidad central de proceso** es la encargada de llevar a cabo las operaciones aritméticas y lógicas, es decir, el procesamiento de la información.

La **unidad de memoria** de sólo lectura contiene el sistema operativo con todos los elementos necesarios para su funcionamiento: archivos de inicio del sistema, drivers[11] y aplicaciones que vienen empotradas de fábrica. Esta memoria mantiene la información con independencia de la alimentación eléctrica y no es accesible al usuario para realizar escrituras.

La **unidad de memoria RAM** contiene varias secciones: la pila de ejecución (stack), el heap (montón) que es la zona donde las aplicaciones almacenan los objetos solicitados de forma dinámica, secciones

11. Los drivers (o controladores) son piezas de software que permiten al sistema operativo la administración de un dispositivo hardware.

reservadas para uso del sistema operativo y el sistema de archivos. Todos los datos, archivos y aplicaciones que incorpore el usuario se almacenan en memoria de ejecución. Sabemos que la memoria RAM es volátil, es decir, si es desconectada de la alimentación pierde toda la información contenida en ella. No obstante, la arquitectura de Pocket PC incluye una batería interna de resguardo que mantiene alimentada la memoria por si ocurren problemas con la batería de operación (puede que se descargue, falle, no esté disponible). La batería interna no es accesible al usuario.

A continuación se muestra la figura 1.3 tomada de [SDK03.1] a fin de visualizar los componentes básicos de un Pocket PC.

Figura 1.3 Componentes de Pocket PC

Pantalla Sensible al Tacto

La pantalla de LCD tiene un tamaño estándar de 240x320 pixels de resolución de orientación apaisada, con tamaños de punto que varían de .22 a .24 mm. según el fabricante. Se puede tocar la pantalla con el stylus o bien con la mano, lo cual tiene un efecto similar al uso del mouse. Soporta una profundidad de color de hasta 16 bits por píxel.

Stylus y Teclado

Pocket PC no tiene un teclado hardware estándar. En cambio, ofrece un teclado emulado por software con disposición QWERTY[12] usando el stylus y reconocimiento de escritura sobre la pantalla.

Botones de Navegación

Se proveen una serie de controles de navegación. Permiten ampliar las opciones de control de las aplicaciones y pueden ser presionados, sostenidos, hacerles doble clic y presionados conjuntamente con otros controles. Entre ellos figuran:

- Tecla de encendido/apagado
- Tecla de acción (Enter)
- Control flecha arriba
- Control flecha abajo
- Teclas de programas (inician aplicaciones)
- Tecla de grabación de voz

CPU

La familia de Pocket PC viene equipada con procesadores marca ARM que priorizan el bajo consumo de energía y la performance.

Memoria

Los equipos actuales incluyen al menos 24 MB de memoria ROM y 16 MB de RAM para la ejecución. A fin de conservar la memoria RAM disponible, algunos componentes del sistema operativo vienen incluidos en la ROM. Algunas manufactureras equipan al dispositivo con puertos para memoria Flash a fin de expandir las capacidades.

Energía

Los equipos llevan una batería a fin de operar con independencia de la red eléctrica durante varias horas. Incluyen internamente una batería de seguridad para evitar la pérdida de datos si la batería principal falla.

12. Disposición de las teclas en teclados estándar de idioma inglés.

Audio

Incluyen speakers para que el usuario pueda oír audio y algunos fabricantes proveen una entrada de audio para conexión a otros equipos de audio.

Puerto Serie

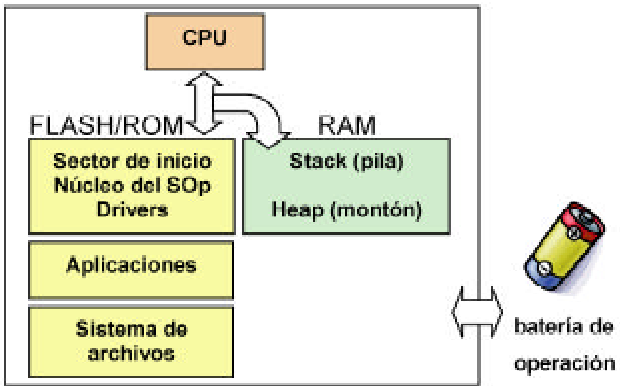
Incluye un puerto serie de conexión cuya velocidad de transferencia varia entre 19.2 Kbit/segundo hasta 115Kbit/segundo. Este puerto está pensado para la conexión del dispositivo a una computadora de escritorio o a otros dispositivos. Algunos fabricantes proveen un puerto USB en vez del serie.

Puerto Infrarrojo

Pocket PC incluye en puerto de comunicaciones infrarrojo que se adapta a las especificaciones Infrarred Data Association (IrDA).

Arquitectura Smartphone

Figura 1.4 Arquitectura de Smartphone



La figura 1.4 muestra la arquitectura de un dispositivo Smartphone. Existen tres componentes principales:

· la unidad central de proceso (CPU)
· una unidad de memoria de almacenamiento permanente (puede ser tecnología Flash)
· una unidad de memoria RAM

Al igual que en Pocket PC, la **unidad central de proceso** es la encargada de llevar a cabo las operaciones aritméticas y lógicas, es decir, el procesamiento de la información.

La **unidad de memoria de almacenamiento permanente** contiene el sistema operativo con todos los elementos necesarios para su funcionamiento: archivos de inicio del sistema, drivers y aplicaciones que vienen empotradas de fábrica. La diferencia fundamental con Pocket PC es que en este dispositivo el sistema de archivo (por ende archivos de usuario) reside en esta memoria no volátil; no es necesaria una batería de resguardo. Por último, la **memoria RAM** es exclusivamente para la ejecución del sistema operativo y aplicaciones.

La figura 1.5 tomada de [SDK03.0] muestra los componentes de hardware básicos:

Figura 1.5 Componentes básicos de Smartphone

CPU

Los equipos cuentan con un procesador de 32 bits.

Memoria

Poseen un mínimo de 4MB de RAM dinámica y un mínimo de 8MB de ROM o tipo Flash.

Pantalla

No posee características de sensibilidad al tacto. Las dimensiones estándar son 176x220, 160x240 y 208x240 píxel, con un tamaño de punto de 0.20 mm., orientación apaisada y profundidad de color de 8 o 16 bits por píxel para pantallas color.

USB

Incluye soporte Universal Serial Bus cliente hasta 12Mbits/segundo

Teclado

Incluye un conjunto de teclas para:

- encendido
- flechas de desplazamiento (arriba, abajo, izquierda y derecha)
- botón de acción
- volver atrás
- enviar
- detener
- dígitos alfanuméricos, *, #
- tecla de grabación de voz.

Energía

Baterías recargables con autonomías de acuerdo a los niveles requeridos.

Audio

Incluye micrófono, speaker y hardware de grabación. Algunos fabricantes proveen conectores de audífonos externos.

Diferencias Principales entre Pocket PC y Smartphone

Tabla 1.1 Diferencias Principales entre Pocket PC y Smartphone

	Pocket PC	**Smartphone**
Procesador	Procesadores actuales hasta 400 MHz.	Menor capacidad de procesamiento, actualmente hasta 120 MHz.
Memoria RAM	memoria estándar de 64 y 128 MB	Menor cantidad de memoria RAM, aproximadamente 16 MB
Memoria ROM	entre 32 y 48 MB	
Memoria FLASH	opción de tarjetas externas	64 MB, 32 disponibles
Sistema de archivos	en RAM con batería de seguridad	en memoria Flash, mas lento que RAM
Display	tamaño estándar de 240x320 píxeles (TFT)	tamaño estándar de 176x220 píxeles (TFT)
Entrada	pantalla sensible al tacto stylus teclado software teclas hardware de acción grabador de voz	teclado alfanumérico teclas hardware de acción tecla por software (softKey)
Peso aproximado	90-200 gramos	80-150 gramos
Autonomía Baterías	4-12 horas de uso	4-7 horas de habla 100-170 horas en inactividad
Adm. de la energía	Modo de *inactividad* (standby): Cuando no se utiliza, se apaga completamente el sistema operativo y lo único que se mantiene encendido es el CPU en espera de la activación de la tecla «Power» (encendido)	Modo ocioso (idle): A diferencia de PPC, necesita estar siempre encendido por ser un teléfono celular. El sistema operativo siempre está ejecutándose para mantener la pantalla funcionando, aún cuando el usuario no realiza ninguna acción. No obstante, existe un modo de ahorro de energía.
Cierre de aplicaciones	Las aplicaciones proveen un control para que sean finalizadas como en Windows de escritorio.	No existe un control de cierre de aplicaciones. Simplemente, se ejecuta otra opción. Las aplicaciones se cierran automáticamente en base a la disponibilidad de memoria. *Esto requiere un diseño especial del comportamiento de las aplicaciones.*

Adecuación de los dispositivos al contexto móvil

Los dispositivos móviles están pensados para ser incorporados a las actividades diarias de las personas que los necesitan. Sus características físicas y de diseño determinan el grado de adecuación que ofrecen. Un equipo «portátil» que pesa 3 Kgs. es de esperar que sea rechazado por algunos usuarios, dada la falta de practicidad de traslado que presenta. A continuación se muestran dos tablas con valores *aceptables* e *ideales* para algunas variables que indican el grado de usabilidad, para Pocket PC y Smartphone respectivamente: [MS03.1]

Tabla 1.2 Valores de adecuación para Pocket PC

Volumen del dispositivo	aceptable: ideal:	hasta 135 cc. hasta 110 cc.
Peso	aceptable: ideal:	hasta 170 gr. hasta 140 gr.
Autonomía de uso efectivo (procesamiento)	aceptable: ideal:	8 horas 15 horas
Autonomía en modo de inactividad	aceptable: ideal:	100 horas 150 horas

Tabla 1.3 Valores de adecuación para Smartphone

Volumen del dispositivo	aceptable: ideal:	hasta 100 cc. hasta 80 cc.
Peso	aceptable: ideal:	hasta 110 gr. hasta 80 gr.
Autonomía de uso efectivo (procesamiento sin uso de radio)	aceptable: ideal:	8 horas 12 horas
Autonomía en modo de inactividad	aceptable: ideal:	100 horas 150 horas
Autonomía de uso del teléfono celular (hablando)	aceptable: ideal:	3 horas 4 horas

Limitaciones generales de los dispositivos móviles

Se presenta la siguiente lista a modo de ofrecer al lector una idea general de las restricciones que poseen algunos equipos:

- Display de tamaño pequeño, el área de trabajo es crítica
- No poseen teclado (generalmente)
- Si poseen teclado es pequeño y de funcionalidad reducida
- Los procesadores son menos poderosos que los PC de escritorio y notebooks
- Tienen poca memoria RAM
- No tienen medios de almacenamiento magnético/óptico de alta capacidad
- Las baterías tienen duración muy limitada
- La actualización del software no puede hacerse directamente a través de periféricos como disquete, CD, etc. Se requieren enlaces a una PC.
- Es limitada la oferta de periféricos
- Hay muchos periféricos que no se han desarrollado para estos equipos: scanner
- La escalabilidad de hardware es limitada
- No dan posibilidad de opción del sistema operativo
- La conexión a periféricos no portátiles esta supeditada a la existencia de los conectores físicos y de los controladores para el sistema operativo.
- No son adecuados para la ejecución de software de graficación

2.1 Conceptualización del Ambiente de desarrollo de aplicaciones móviles con Compact Framework

Para desarrollar aplicaciones móviles debe hacerse la elección del ambiente de desarrollo. Cada ambiente se forma mediante una configuración particular de metodologías, modelos de programación, herramientas, documentación de apoyo a la implementación y hardware. Una **metodología** es un conjunto de procedimientos que permiten producir y mantener un producto de software. La metodología define *fases* del ciclo de vida del software de las que se va a ocupar. Los **modelos de programación** consisten en estilos y técnicas de implementación de acuerdo a tecnologías o plataformas particulares. Entre las **herramientas** están los lenguajes, compiladores, depuradores (debuggers), emuladores, etc., idealmente contenidos en un entorno integrado de desarrollo. Si no se hallan integrados en un entorno único, cada herramienta funciona de manera aislada y la integración requiere un esfuerzo adicional que implica tiempos y costos extras. La **documentación** de desarrollo suele presentarse bajo la forma de «kits» que contienen documentos electrónicos con guías, prácticas habituales e información acerca de los modelos de programación. El **hardware** está formado por los dispositivos móviles objetivo para los cuales se desarrolla y por los equipos de escritorio necesarios para ejecutar los entornos de desarrollo. El proceso de implementación en sí ocurre en los equipos de escritorio. La figura 2.1 a continuación representa el gráfico general de un ambiente de desarrollo de aplicaciones móviles.

Figura 2.1. Ambientes de desarrollo de aplicaciones móviles

En esta tesina se propone incorporar el uso de un marco de trabajo al ambiente de desarrollo de aplicaciones móviles. Un marco de trabajo provee idealmente tres elementos: [CP00]

1. un envoltorio (algún nivel de abstracción)
2. una arquitectura
3. algún método

Un **envoltorio** ofrece la abstracción necesaria para simplificar al desarrollador el acceso a las tecnologías de base. Puede materializarse, por ejemplo, a través de un lenguaje intermedio independiente de plataforma de software/hardware, una librería de funciones que den acceso consistente a los servicios del sistema operativo, una API, etc.

La **arquitectura** es la manera en que se relacionan los componentes y tiene un diseño específico. El entorno que controla las condiciones de ejecución del código, la especificación de los mecanismos de interacción de componentes, la especificación de cómo llevar a cabo la reusabilidad, las librerías de clases que ofrecen funcionalidad ya desarrollada, son posibles elementos que, juntos, conforman la arquitectura.

Un **método** es la manera en que se lleva a cabo el desarrollo. Refuerza la adopción de una manera consistente de desarrollar aplicaciones. Puede emplearse, por ejemplo, el método de orientación a objetos y programar para lograr la reusabilidad. Considero que un marco de trabajo puede definir métodos, pero éstos se aplican siguiendo una metodología aparte que no forma parte del marco.

El marco de trabajo propuesto es el *Microsoft Compact Framework* y será definido en las siguientes secciones de este mismo capítulo. De ahí en adelante, todos los temas tratados se basarán en el funcionamiento de este marco.

Herramientas para el desarrollo de aplicaciones móviles

Las herramientas que se proveen para el desarrollo de aplicaciones móviles siguen en general alguno de los siguientes esquemas de codificación [MS03.6]:

· Uso de código nativo. El código fuente es compilado y se genera código de máquina de una plataforma

específica. Se utiliza para aplicaciones que requieren acceso directo al hardware, código máquina de tamaño reducido y alta performance.

- Uso de código administrado[13]. El código fuente es compilado a un lenguaje intermedio que es administrado por un gestor de ejecución. Se utiliza para aplicaciones que requieren un desarrollo rápido, seguro o existe la necesidad de introducirlas rápidamente en el mercado.
- Uso de código de servidor. Se trata de un conjunto simple de código para dispositivos móviles que poseen un ancho de banda garantizado.

Hasta el momento, la empresa desarrolladora del Compact Framework ha propuesto una serie de herramientas para el desarrollo en implementación de aplicaciones móviles. Las herramientas más usuales han sido:

eMbedded Visual C++ 3.0/4.0:

- Genera código nativo
- Óptimo para la producción de drivers o software que opera con el hardware
- Debe interactuar con la API de la plataforma
- Adecuado para aplicaciones que requieren recursos intensivos de procesamiento
- Programación de juegos de alta performance

eMbedded Visual Basic 3.0:

- Genera código que es interpretado.
- Facilita el diseño de interfaces gráficas
- Requiere un módulo de ejecución especial llamado Visual Basic Runtime
- La empresa desarrolladora recomienda que los nuevos desarrollos sean hecho en Visual Basic .NET y que esta herramienta se use para mantenimiento de aplicaciones ya hechas.

Visual Studio .NET 2003: un entorno integrado de desarrollo para .NET

La idea central es proveer un entorno integrado de desarrollo que tenga las mismas herramientas, lenguajes y modelos de programación para aplicaciones de escritorio, de servidor y móviles. De esta forma, no sería necesario un reentrenamiento completo[14] de los desarrolladores que ya cuentan con experiencia, lo cual es muy costoso. Así, los programadores pueden emplear la experiencia ya adquirida en los lenguajes .NET y en el uso de las herramientas de desarrollo. La forma de abordar la construcción de los distintos tipos de aplicaciones es mediante el uso de **proyectos**. Se proveen varios tipos de proyectos y el ambiente de desarrollo adapta sus características, documentación y opciones disponibles para cada proyecto.

Las características principales que ofrece el entorno son: [MS03.5]

- Todos los lenguajes comparten el mismo diseñador visual, interfaz de edición de código inteligente[15], acceso a compiladores/debuggers[16] y herramientas de diseño de bases de datos.
- Existen las mismas facilidades de acceso a datos para todos los proyectos, mediante ADO .NET. Se basa completamente en XML y permite trabajar con datos locales y remotos.
- Permite la integración de sistemas heterogéneos mediante el uso de servicios Web basados en XML, usando protocolos abiertos de Internet.
- Ofrece emuladores específicos de dispositivos móviles. Esta característica es muy útil ya que el desarrollador no requiere hardware adicional para realizar el desarrollo de una aplicación móvil. Los emuladores que se proveen son muy fieles a los dispositivos porque se ejecuta el sistema operativo real del dispositivo objetivo en un proceso local, es decir, no es una simulación. La parte que se simula es la interfaz gráfica de usuario que incluye la pantalla y las teclas hardware de los dispositivos.
- En la versión 2003 se incorporó un conjunto de herramientas de gran utilidad denominadas Smart Device Programmability (SDP), más conocido como **Smart Device Extensions (SDE)**. Tiene el fin de dar soporte a los desarrolladores que utilizan el Compact Framework y es la parte que permite que los desarrolladores puedan generar aplicaciones móviles de forma similar al desarrollo de aplicaciones de Windows de escritorio. Cuando el programador desea probar sus aplicaciones existe la posibilidad de ejecutarlas sobre el emulador o sobre el dispositivo real. Si se cuenta con el dispositivo físico, el entorno

13. Este concepto será ampliado posteriormente en este capítulo por ser el esquema adoptado por el Compact Framework
14. Sin embargo, los desarrolladores de aplicaciones móviles deben conocer y adoptar algunas nuevas prácticas de diseño e implementación para hacer frente a las restricciones actuales que poseen algunos dispositivos. Estos temas se tratan en el capítulo 3.
15. Indica palabras reservadas, muestra la sintaxis de métodos e información de métodos y clases, y reduce la cantidad de texto que debe tipear el usuario.
16. El proceso de debugging consiste en la depuración de los errores que contenga la aplicación. El entorno ofrece facilidades como inspección de variables, punto de corte en la ejecución y ejecución paso a paso.

de desarrollo determina automáticamente qué archivos y recursos necesita, y comienza la transferencia de datos. Concluida la transferencia, puede iniciarse el proceso de debugging.

- La función de depuración remota permite realizar la corrección del código para ejecución local emulada o si es realizada en el dispositivo físico.

ASP.NET

Como complemento del *Smart Device Extensions* existe el *Microsoft Internet Mobile Toolkit* (MMIT). Es una extensión del Compact Framework y del Visual Studio para el desarrollo de aplicaciones Web móviles. Permite que los diseñadores utilicen Visual Studio para que las aplicaciones Web móviles adapten su forma de presentación automáticamente de acuerdo al dispositivo en que se ejecuten. ASP no instala componentes en el dispositivo móvil, lo que hace es producir código ejecutable en el servidor para adaptar la disposición en pantalla a los distintos navegadores Web. Genera código HTML, Compact HTML y Wireless Markup Language (WML).

La tabla 2.1 a continuación muestra las herramientas que ofrece la empresa desarrolladora del Compact Framework para las distintas opciones de entornos de ejecución de código, mientras que la tabla 2.2 muestra las herramientas que soporta cada plataforma y cuáles de ellas han sido las más utilizadas por los desarrolladores.

Tabla 2.1 Herramientas disponibles según tipo de código

	Herramientas Microsoft	
código nativo	eMbedded Visual Tools 3.0: Visual C++	eMbedded Visual C++ 4.0
código interpretado	eMbedded Visual Tools 3.0: Visual Basic 3.0	eVB Runtime
código de servidor	ASP .NET	
código administrado	Compact Framework	Visual Studio .NET

Tabla 2.2 Herramientas soportadas por cada plataforma móvil

	eMbedded Visual C++ 3.0	eMbedded Visual Basic	eMbedded Visual C++ 4.0	Compact Framework	ASP .NET
Pocket PC/2002	✍	✍		✍	✍
Smartphone 2002	✍				✍
Pocket PC 2003			✍	✍	✍
Smartphone 2003			✍	✍	✍

✍ Indica que la herramienta está soportada por la plataforma

Indica que ha sido la herramienta más utilizada

2.2 El marco de trabajo de Microsoft .NET Compact Framework

Presentación del .NET Compact Framework

La idea central de .NET es un conjunto de **software** para conectar personas, sistemas, información y dispositivos a través del uso de XML Web Services.

El marco de trabajo .NET compacto proviene del marco de trabajo .NET estándar. Ambos pueden ser definidos en base a sus dos componentes principales, un entorno que administra la ejecución de las aplicaciones y provee abstracción respecto de su plataforma subyacente y por otra parte, una extensa librería usada por los desarrolladores para construir las aplicaciones, la .NET Base Class (FCL, Framework Class Library). FCL representa la API[17] para el Framework y por ello todas las implementaciones .NET soportan la librería.

Debe quedar claro que el Framework no es un sistema operativo, Windows es el sistema operativo y su API sigue ejecutándose de manera indirecta, es decir, no es el usuario quien invoca a la API directamente

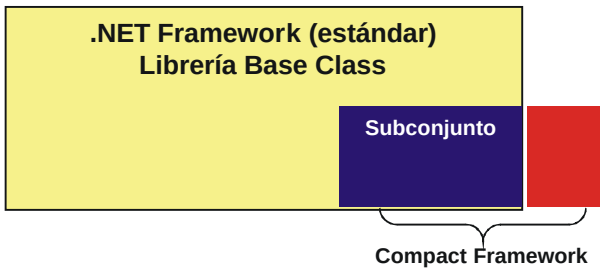
17. En inglés, Application Programming Interface, que permite a los usuarios utilizar primitivas de programación del sistema operativo.

sino el gestor de ejecución. El gestor, llamado Common Language Runtime (CLR) se sitúa en una capa intermedia entre el sistema operativo y las aplicaciones.

El Compact Framework es una versión reducida respecto del Framework estándar, compuesto por:

a. un subconjunto de las librerías que forman la .NET Base Class
b. un superconjunto que incorpora a la librería nuevas clases que son de utilidad para el desarrollo de aplicaciones en dispositivos móviles
c. una nueva implementación del CLR

Figura 2.2 .NET Framework Base Class Library Vs .NET Compact Framework



¿Porqué incluye un subconjunto de las clases que forman la Base Class Library?

Como ya se ha tratado anteriormente, los dispositivos móviles cuentan con importantes restricciones en cuanto a su capacidad de memoria para ejecución y de almacenamiento. Dado que el Framework es requerido para la ejecución de todas las aplicaciones .NET, es de esperar que éste no ocupe demasiado espacio en relación a las capacidades de los dispositivos. No obstante, la implementación del Compact Framework es muy completa y reducida[18] en tamaño a la vez.

Omite las clases que no agregan utilidad directa en el dominio de las aplicaciones móviles o aquellas que pueden contar con una implementación alternativa que representen menor requerimiento de espacio de almacenamiento. Específicamente, se excluyeron aquellas clases que exponen servicios disponibles en computadoras con Windows de escritorio y dichos servicios no están soportados por Windows CE[19], aquellas clases que implementan funciones no usadas comúnmente en dispositivos móviles y aquellas demasiado costosas de implementar computacionalmente. El marco reducido excluye selectivamente, además, algunas clases, métodos, enumeraciones e interfaces.

A continuación se expone una síntesis de algunas características excluidas:

Tabla 2.3 Algunas características excluidas en .NET Compact Framework

Característica	Descripción
Serialización	BinnaryFormater se utiliza para la serialización binaria de objetos y SoapFormatter para la serialización con el protocolo SOAP. Se han excluido por consideraciones de tamaño y performance, no obstante existen alternativas para la serialización utilizando XML.
Soporte para la impresión	Se piensa que los dispositivos móviles suelen ser para la recolección de datos y consulta de información. La impresión ha sido de interés sólo para una minoría de usuarios y además el sistema operativo PocketPC no la soporta. Esta característica ha sido librada a terceros proveedores.
Remoting	La característica de comunicarse con componentes remotos en otras máquinas puede ser implementada utilizando XML Web Services

18. Tiene un tamaño de 1,5 MB aproximadamente (referencia de compilación en un procesador marca ARM, ampliamente utilizados en PDA)
19. Windows CE es un sistema operativo de la compañía Microsoft pensado para dispositivos que llevan software empotrado en general, entre ellos algunos dispositivos móviles.

Formularios Web	Las clases que integran el espacio de nombres *System.Web,* incluyen características de servidor. El servicio de contenidos Web no se considera parte del dominio de las aplicaciones móviles, dado que éstas se comportan generalmente como clientes.
Interoperabilidad COM	Consiste en permitir a las aplicaciones invocar a objetos COM, probablemente heredados. Se ha excluidos por razones de espacio de almacenamiento.
Desplegar Internet	Una práctica común es desarrollar aplicaciones cuya única función consiste en bajar de Internet la última versión de una assembly[20] .NET y ejecutarla en el cliente. Esta modalidad presupone la existencia de una conexión consistente a la red, lo cual no ocurre en el caso de dispositivos móviles. Además el marco de trabajo compacto no dispone de caché para las bajadas de archivos.
Criptografía	Esta característica, es si duda, una de las más importantes en el ámbito de las aplicaciones móviles y aún así ha sido excluida. Encontramos aquí una deficiencia importante que requiere una solución alternativa. No surge precisión de porqué no ha sido incluida en el marco, pero presupongo que se debe a cuestiones de performance. La encriptación requiere importantes recursos de procesamiento y almacenamiento.

¿Porqué se han incorporado nuevas clases al Compact Framework?

El Framework completo no incluye en su librería de clases utilidades especiales para el desarrollo de aplicaciones móviles.

Los gestores de bases de datos para equipos de escritorio usualmente requieren grandes espacios de almacenamiento, por ello se desarrollaron versiones especiales de servidores de bases de datos como SQL Server CE[21]. Para que las aplicaciones móviles .NET hagan uso de los datos, se incluyeron en la librería clases especiales para el manejo de datos en este servidor. Dicha librería es *System.Data.SqlServerCe.* Por otro lado, para dar soporte a comunicaciones infrarrojas se crearon clases para las ya existentes librerías *System.Net* y *System.Net.Sockets*.

Para dar soporte al software input panel (SIP[22]) y a la comunicación entre procesos también se han incorporado nuevas clases.

¿Porqué provee una nueva implementación del CLR?

En vista de las limitaciones de los equipos móviles expuestas en el Capítulo 1, el entorno de ejecución de aplicaciones .NET debió adaptarse por razones de:

- performance
- optimización de recursos
- ahorro de energía para maximizar la utilización de las baterías

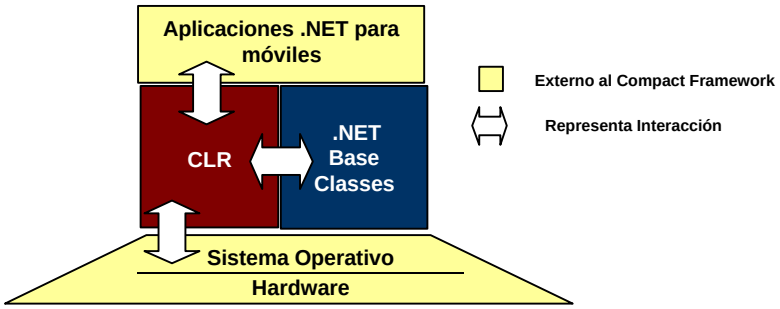
Arquitectura del Compact Framework

La figura 2.3 muestra cómo se organiza el marco de trabajo en base a sus dos componentes principales. El bloque de aplicaciones (externo al marco de trabajo) representa el código intermedio MSIL de cualquier aplicación .NET. El código será ejecutado y gestionado por el Common Language Runtime a fin de aislar la

20. Concepto que más adelante se desarrollará. De forma sintética, es una unidad lógica que incluye el código intermedio de la aplicación e información adicional para la correcta gestión de la misma (metadatos).
21. SQL Server CE es una versión reducida de SQL Server de la compañía Microsoft, para el uso en dispositivos móviles de capacidades limitadas.
22. Es la forma es que los PDA ingresan sus datos, por medio de un teclado emulado por software.

aplicación de otras que puedan interferir en su ejecución normal, crear condiciones de seguridad y proveer independencia del sistema operativo de fondo. La librería de clases aporta funcionalidad extra a las aplicaciones y un marco consistente de orientación a objetos.

Figura 2.3. Ejecución de aplicaciones en Compact Framework



El uso de las clases ya incorporadas al Framework es una de las ventajas mayores porque facilitan notablemente al programador la tarea de implementación y de acceso a servicios del sistema operativo.

La .NET Base Class es una gran colección de clases de código administrado, las cuales son provistas por la empresa creadora del Framework y permiten al desarrollador hacer cualquier tarea que anteriormente se hacía con el API de Windows.

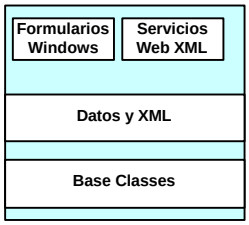
Las clases se organizan según un esquema de herencia simple y siguen el mismo modelo de objetos que utiliza el lenguaje intermedio, por esto es posible instanciar una clase de la librería o derivarla en una nueva que incorpore funcionalidad adicional. Lo más destacable de esta jerarquía de clases es su relativa facilidad de uso y su gran cobertura del API de Windows. A continuación se muestra una tabla con algunas áreas de cobertura de las clases en el Compact Framework:

Tabla 2.4 Algunas áreas de cobertura de clases en el Compact Framework

Área	Espacio de nombres que la implementa
Diseño de formularios tipo Windows	*System.Windows.Forms*
Elementos para graficación en 2D	*System.Drawing*
Soporte para XML	*System.Xml*
Acceso a datos	*System.Data*
Servicios de **cliente** Web	*System.Web*
Seguridad	*System.Security*
Conectividad	*System.Net*
Colecciones de datos	*System.Collections*
Servicios de tiempo de ejecución	*System.Runtime*
Manejo de Texto	*System.Text*
Entrada y salida	*System.IO*

En [MS01.1] se propone una organización esquemática de las librerías de clases del Compact Framework:

Figura 2.4 Distribución de las clases en Compact Framework



- Formularios Windows: contiene las clases necesarias para construir GUIs[23].
- Servicios Web XML: contiene clases para que la aplicación móvil pueda ser cliente XML de servicios web.
- Datos y XML: están las clases para acceso a datos y para la conversión de contenedores de datos ADO.NET a XML y viceversa.
- Base Classes: incluye los tipos de datos primitivos, estructuras de datos de uso frecuente como *arrays*, funciones de entrada y salida, seguridad y funciones para globalizar las aplicaciones.

El procedimiento genérico para el desarrollo de una aplicación en esta arquitectura es el siguiente: los desarrolladores eligen un lenguaje soportado por el marco de trabajo y construyen sus aplicaciones. Al ser compiladas se transforman en MSIL[24], que es una representación intermedia independiente de la plataforma de hardware y software subyacente. El linker[25] empaqueta este código junto a las referencias hechas a las Base Classes y posteriormente es recompilado a código de máquina para ser ejecutado en cualquier plataforma que provea una implementación del Framework.

El gestor de tiempo de ejecución (CLR) y código administrado

La unidad de ejecución del Compact Framework es la ***assembly***, una estructura lógica que resulta de la compilación y enlace del código. Posee carácter lógico porque puede estar contenida en uno o más archivos físicos de tipo *.dll* o *.exe.* Tiene una cierta similaridad con las antiguas DLLs, pero esta estructura es completamente autodescriptiva, pues, incluye el código MSIL de la aplicación junto a metadatos[26] que describen la assembly. Los metadatos se guardan en una sección especial denominada *manifiesto*[27]*,* que contiene información acerca de los tipos[28] de los miembros[29], y referencias a otras assemblies. Con esta información complementaria, el gestor de tiempo de ejecución tiene instrucciones específicas acerca de cómo ejecutar el código.

Las assemblies facilitan, además, el control de versiones de componentes de software por medio de números de identificación que se guardan en el manifiesto.

Una vez creada la assembly hemos pasado la primera instancia de compilación pero aún no se ejecuta la aplicación. Es a partir de ahora donde el CLR toma un rol protagónico. Se encarga de:

- cargar una assembly (código ya compilado a MSIL)
- crear un dominio de aplicación[30] para que se ejecute el código contenido en la assembly
- utilizar un compilador JIT para compilar los métodos a código nativo que correrán en el procesador local
- administrar el recurso de memoria: alocación y recolección de basura[31]
- garantizar condiciones de seguridad: tratamiento de excepciones, permisos, etc.

Vamos ahora a explicar por partes la función del Common Language Runtime.

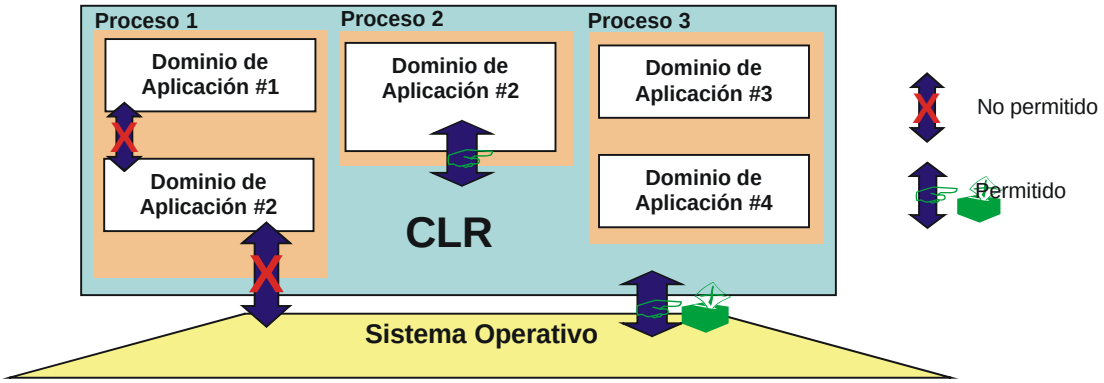
Carga de assemblies y creación de dominios de aplicación:

Situándonos anteriormente a la tecnología de .NET, los *procesos* [STA01] se utilizaban como unidades de aislamiento, es decir, cada proceso contaba con su imagen de memoria para el código, pila de ejecución y datos. Por lo tanto, una aplicación se ejecutaba en un proceso y no podía escribir en la imagen de memoria de otro proceso y consecuentemente corromper la aplicación. El encargado en este modelo de trabajo era el sistema operativo, quien para cada aplicación asignaba un proceso de ejecución. Con la arquitectura .NET surge un nuevo límite para las aplicaciones, el «**dominio de aplicación**». En un entorno con código administrado por el CLR se puede asegurar que una aplicación no podrá acceder al área de memoria de otra aplicación, aunque se ejecuten en el mismo proceso. De esta forma, múltiples aplicaciones pueden correr en un proceso simple pero cada una lo hará en su dominio de aplicación. El control del dominio de aplicación es exclusivo del CLR. El modelo de los dominios de aplicación puede verse de manera esquemática mediante la siguiente gráfica:

---

23. En inglés, Graphic User Interface que significa interfaz gráfica de usuario.
24. En inglés,
Microsoft Intermediate Language que significa lenguaje intermedio Microsoft.
25. La función del linker (enlazador) es crear una unidad de carga que consiste en programas y datos, para que el loader (cargador) del CLR los lleve a memoria de ejecución.
26. Información sobre los propios datos.
27. Traducción al español de la denominación Microsoft «manifiest»
28. [GUE98] Ghezzi define el concepto de tipo como la especificación del conjunto de valores que pueden ser asociados a un miembro, junto a las operaciones válidas usadas para crear, acceder y modificar sus valores.
29. Métodos y variables de las clases
30. Traducción al español de la denominación Microsoft «application domain». Es el medio por el cual el CLR permite a códigos distintos ejecutarse en el mismo espacio de un proceso.
31. Mecanismo de administración de la memoria RAM que libera de forma automática y devuelve al sistema operativo el espacio en memoria solicitado previamente y que ya no es utilizado por la aplicación.

Figura 2.5. Modelo de Dominios de Aplicación



En comparación a los procesos estándar, los dominios de aplicación tienen ventajas adicionales. Una ventaja del uso de dominios de aplicación es que éstos pueden crearse dinámicamente, de manera que es posible parar la ejecución de una aplicación sin frenar al proceso que la contiene. La activación y terminación de aplicaciones es mucho más económica computacionalmente que si residieran en procesos estándar. El CLR evita las llamadas entre objetos que pertenecen a distintos dominios de aplicación.

El nuevo concepto de dominios de aplicación significa que diferentes aplicaciones, que necesitan ser aisladas unas de otras pero además necesitan mecanismos de comunicación entre ellas, pueden ser ubicadas en el mismo proceso, con grandes beneficios de performance.

De manera predeterminada, hay un solo dominio de aplicación por proceso. Puede crease uno nuevo con la clase *System.AppDomain.ExcecuteAssembly*.

La actual implementación del cargador de Windows CE (loader) lanza un nuevo proceso y CLR para cada aplicación en vez de chequear la existencia previa de un proceso que esté ejecutando el CLR.

Las aplicaciones que corren bajo el control del CLR se dicen «administradas» por él, de aquí surge el concepto de **código administrado,** en oposición al código que se ejecuta por debajo del CLR accediendo directamente a los servicios del sistema operativo, el cual se denomina **código no administrado.**

El entorno de ejecución del Compact Framework puede dividirse conceptualmente en dos secciones: la que trata con código administrado y la que trata con **código nativo**[32]. Nótese que el código nativo es código no administrado y no necesariamente la inversa. En la primera sección, las aplicaciones, las librerías de clases específicas y la mayoría de las clases FCL son compiladas a MSIL para ejecutarse bajo el control del CLR.

En la segunda sección reside el motor de ejecución, que en sí es código nativo ejecutable al igual que la capa de abstracción de la plataforma subyacente, PAL[33]. Esta capa y el motor de ejecución han sido implementados conjuntamente en un archivo llamado Mscoree.dll.

Es importante aclarar que el concepto de **código no seguro**[34] no es sinónimo de código no administrado. La diferencia radica en que el código no seguro es etiquetado como tal para que el CLR lo identifique y permita ejecutar acciones que «contravienen los principios de seguridad y la buenas prácticas de programación en .NET», pero aun así se ejecuta bajo el control del CLR. El código no administrado escapa al ámbito de acción del CLR.

Compilador JIT[35]

Hay dos instancias de compilación en una aplicación .NET. La primera ocurre al código fuente, el cual es traducido a MSIL, antes de la ejecución. La segunda es en tiempo de ejecución, tras la carga de las assemblies. Se traduce el código MSIL a código nativo. Esta segunda compilación es producida por el JIT *compiler*.

El código administrado debe atravesar un proceso de **verificación** antes de ser ejecutado. La verificación garantiza que una aplicación no accede a las direcciones de memoria de otras por medio de la validación de tipos. Antes de que cualquier método pueda ejecutarse, el compilador JIT produce código nativo. En este compilador se halla el verificador que asegura que todos los tipos en el código han sido declarados y

32. Se refiere al código de las aplicaciones en lenguaje de máquina, dependiente de la plataforma de hardware.
33. En inglés, Platform Adaptation Layer que es una capa de abstracción entre el motor de ejecución y el sistema operativo.
34. Traducción al español de la denominación Microsoft «unsafe code».
35. JIT es el acrónimo para Just-In-Time o justo a tiempo. Indica realización bajo demanda.

operados adecuadamente. Es consecuencia de esto que una aplicación .NET no puede fallar a causa de conversiones de tipo inseguras, variables no inicializadas, desbordamientos de buffers o indexación de vectores fuera de límites. Si el CLR no puede verificar las condiciones de seguridad de tipo, simplemente rechazará la ejecución.

Validado ya el código, el CLR lo ubica para que se ejecute en un dominio de aplicación. Toma la primera porción de código, lo compila y lo ejecuta en un thread[36] apropiado del programa. Cada vez que el CLR encuentra una referencia a un nuevo método, éste es compilado a código nativo y ejecutado. La compilación de cada método ocurre sólo una vez sólo cuando es referenciado, pues referencias sucesivas al mismo método quedan almacenadas en una memoria caché del CLR.

Este proceso tiene un beneficio especial para el Compact Framework, ya que reduce el espacio de almacenamiento requerido en dispositivos móviles: sólo se compilan a código nativo los métodos que se utilizan y el código restante MSIL es mucho menor en tamaño que el nativo. Cuando ya no se utiliza el código nativo, es descartado liberando el espacio en memoria.

Administración de la Memoria en el CLR

El recolector de basura (**garbage collector**) es el principal mecanismo de .NET para la administración de la memoria de ejecución. Hasta ahora han habido dos enfoques previos usados en sistemas Windows para administrar el uso de la memoria que se solicita dinámicamente: el primero consiste en escribir código adicional para liberar recursos de memoria manualmente, el segundo, mantener objetos que hagan conteos de referencias. Esta última técnica consiste en incrementar un contador de referencias cada vez que un objeto en memoria es referenciado por algún otro objeto, y decrementarlo cada vez que un objeto deja de referenciarlo. Cuando ya no existen referencias a un objeto, se lo elimina de la memoria. El primer enfoque tiene la ventaja de ser muy eficiente pero suele ser complejo de programar y propenso a errores, y el conteo de referencias es muy costoso computacionalmente. Las técnicas mencionadas son aceptables en entornos donde el código de la aplicación accede directamente a los servicios del sistema operativo, pero en un ambiente de código administrado la gestión de memoria puede hacerse de manera simple y segura.

El garbage collector de .NET se basa en el principio que la memoria solicitada dinámicamente se halla en el *heap*[38]. Cuando el CLR detecta que el heap de un proceso se está ocupando en un alto porcentaje, llama al garbage collector. Éste recorre las variables del código que se encuentran en un determinado ámbito examinando si referencian a objetos ubicados en el heap. Si no halla referencias activas, supone que ya no son accesibles por el código y por lo tanto debe ser liberada la memoria que las variables utilizaban en el heap.

Este proceso es no determinístico, es decir, no sabemos el momento exacto en que la recolección ocurrirá; la tarea queda en manos del CLR.

La recolección se ve facilitada por la forma en que ha sido diseñado el MSIL, pues es de tipo seguro: una referencia alcanza para saber el tipo del objeto al que apunta.

Es simple, porque el programador no necesita proveer código adicional ni destructores[39] para liberar recursos de memoria solicitados, se hace de forma automática. Y es segura, porque es realizada por el CLR que maneja código cuyos tipos han sido comprobados, no queda en manos de las habilidades del programador para evitar errores.

Condiciones de Seguridad en el CLR

Ya hemos visto que los dominios de aplicación son una primera instancia para procurar la seguridad de las aplicaciones. No obstante, el acceso irrestricto a memoria no es la única causa de problemas. Puede ocurrir que el código escrito por un desarrollador no contemple casos en que su aplicación tiene un comportamiento inesperado. Designaremos con el término **excepción**[40] a la interrupción del flujo de ejecución normal de una aplicación, que se da de manera no deseada, y a veces, inesperada. Esto puede ser consecuencia de una operación aritmética no válida, el acceso a un archivo no disponible, una asignación que no

36. Thread (o hilo) es una unidad de ejecución independiente contenida en un proceso.
37. El heap (o montón) es un área reservada de un proceso para la alocación de memoria solicitada dinámicamente.
38. Es el concepto complementario de un constructor. Algunos lenguajes de programación permiten la petición dinámica de memoria y se deben proveer destructores para devolver los recursos al sistema operativo.
39. No indica necesariamente un error de programación, puede deberse también a errores de ingreso de datos por parte del usuario o a condiciones fortuitas.
40. La pertenencia de un lenguaje al Compact Framework está dada por la implementación de su respectiva versión .NET, es decir, se adapta el lenguaje a este marco y se crean nuevos compiladores que lleven el código fuente a MSIL.

cumple las reglas de tipo seguro, etc. Durante la ejecución normal del código pueden ocurrir algunas de las condiciones anteriores y es imposible continuar ejecutando el programa.

La Base Class Library incluye un completo soporte para el tratamiento de condiciones de excepción. Los desarrolladores de aplicaciones .NET pueden hacer uso de estas facilidades para controlar la ejecución de las mismas.

Es el CLR el encargado de manejar y reportar en tiempo de ejecución las ocurrencias de excepciones, a fin de que las aplicaciones puedan realizar acciones previstas de corrección, informar al usuario y continuar su ejecución normal. El mecanismo se plantea de manera simple: se demarca una sección de código que potencialmente podría producir condiciones de excepción y se provee un **manejador** para *tratar* la excepción ocurrida. Este mecanismo provee una manera conveniente de obtener detalles precisos de la excepción ocurrida, y así poder «derivar el caso» a un método especializado que trate la excepción.

El CLR, provee un alto nivel de confianza de que el código a ejecutar es confiable en cuanto al manejo de recursos y tipos. También se ocupa de aspectos relacionados a la seguridad determinados por **políticas**; reglas explícitas que los administradores definen para lograr aplicaciones seguras. El Framework soporta las políticas de seguridad por medio de las assemblies. Todo el código se ejecuta bajo un contexto de seguridad del CLR: el gestor de ejecución toma las políticas de seguridad establecidas por el administrador del sistema y decide qué permisos se conceden al código en cada nivel. Puede controlarse a qué recursos tiene acceso cada usuario o sistema.

Interoperabilidad de lenguajes y desarrollo para varias plataformas

La idea básica de la interoperabilidad de lenguajes es que teniendo clases o métodos escritos en diferentes lenguajes pertenecientes[41] al marco de trabajo .NET, tales componentes puedan comunicarse de forma directa sin la necesidad de software adicional de comunicación de componentes. Con este enfoque, una clase en un lenguaje genérico del Compact Framework podría invocar directamente a métodos o heredarse de una clase en otro lenguaje. También podría utilizarse un módulo escrito en un lenguaje y tratar las excepciones que lanza desde un segundo módulo escrito en otro lenguaje .NET.

El valor agregado de esta característica es un aumento en la ***reusabilidad*** de componentes, pueden compartirse piezas de software escritas en diferentes lenguajes de la plataforma. Para proyectos de mediano y gran tamaño los desarrolladores de aplicaciones móviles pueden dividirse las tareas según sus distintos perfiles y aún así es posible un desarrollo integrado.

Como veremos posteriormente, el Compact Framework es un medio para que el desarrollo de las aplicaciones móviles no sea un área exclusiva de personal especializado. Así, desarrolladores de aplicaciones de escritorio con experiencia previa en la .NET pueden adaptarse rápidamente al desarrollo de aplicaciones móviles reduciendo costos para la organización, tiempo de desarrollo y de lanzamiento al mercado. La cualidad de interoperabilidad de lenguajes de .NET aporta significativamente a estos fines.

¿Qué hace posible la Interoperabilidad de lenguajes en .NET? [WRO01.0]

Son tres los elementos fundamentales de la plataforma que permiten esta cualidad:

- el código intermedio MSIL
- la existencia de un sistema común de tipos
- la existencia de una especificación común de lenguajes

MSIL

Una vez más puede verse que ante situaciones donde se requiere algún nivel de integración de elementos heterogéneos se necesitan estándares, o al menos especificaciones comunes a las cuales atenerse.

El primer elemento que posibilita la interacción de lenguajes es MSIL. Este lenguaje ha sido diseñado de tal forma que, inevitablemente, debe implementar algún método de programación en particular. Consecuencia de ello es que cualquier lenguaje que vaya a ser compilado a MSIL debe tener compatibilidad con el método elegido. El método que los diseñadores de MSIL eligieron es la orientación a objetos clásica, con un esquema de herencia simple. La decisión tomada tiene una implicación muy importante: los conceptos de clases y herencia se definen a nivel de lenguaje intermedio. Aquí se evidencia una parte de la interoperabilidad de lenguajes. Por otro lado, esto genera limitaciones prácticas pues no todos los lenguajes existentes podrían ser compilados a MSIL, al menos sin una adaptación conceptual.

41. En la terminología de .NET, los miembros públicos son aquellos que pueden ser accedidos desde cualquier clase. Los protegidos son aquellos que sólo pueden ser accedidos por la clase que los contiene o por sus clases heredadas. Y los privados son aquellos que solo pueden ser accedidos por la clase que los contiene, su implementación y existencia es oculta para todo el resto.

CTS y CLS

Para que componentes escritos en distintos lenguajes puedan operar entre sí, cada uno debe conocer todos los tipos de datos utilizados por el otro. La cuestión de obtener acceso a la información de tipos de una clase o método es relativamente simple, dado que los metadatos de las assemblies contienen esta información necesaria. Así, un compilador podría recurrir a los metadatos y obtener la información necesaria de tipos para compilar su código. No obstante, el compilador necesita más información que la ofrecida por los metadatos.

El siguiente pilar de la interoperabilidad de lenguajes es la existencia de un sistema común de tipos denominado **CTS o Common Type System**. Este sistema hace que MSIL maneje un conjunto de tipos predefinidos bien identificados, organizados en un esquema jerárquico de herencia simple según las reglas de la programación orientada a objetos.

El CTS provee un sistema de tipos de datos predefinidos disponibles en MSIL, de manera que todos los compiladores de lenguajes .NET deben producir código basado en estos tipos básicos.

> No significa que un lenguaje .NET deba implementar todos los tipos del CTS, ni tampoco que no pueda tener tipos exclusivos. Los tipos exclusivos de un lenguaje que no tengan correspondencia con el CTS no son interoperables.

Un ejemplo simple que evidencia la necesidad de una especificación común de tipos es: una clase escrita en VisualBasic .NET posee un método que recibe como parámetro una variable de tipo *Integer*, y dicha clase es utilizada por un módulo escrito en C#. Surge la pregunta de qué tipo de C# debería de ser el parámetro que será pasado como argumento a la función de VB. Es lógico que ambos tipos deban ser compatibles para que el método funcione adecuadamente. El problema en este ejemplo es que C# no tiene un tipo llamado *Integer,* pero en cambio tiene un tipo llamado *int* cuya especificación coincide exactamente con *Integer*. ¿Quién indica entonces qué tipo perteneciente a un lenguaje se corresponde con los restantes tipos? La respuesta es: el CTS. Cuando ambos programas, C# y VB, sean compilados a MSIL, los compiladores que conocen el CTS harán que las variables de tipo *Integer* e *int* sean declaradas con el mismo tipo *System.Int32*, pues ambos tipos son compatibles.

La estrategia detrás del CTS, como se dijo anteriormente, es que cada lenguaje .NET implemente el sistema común de tipos. Los desarrolladores que pretenden manejar componentes interoperables deben usar sólo los tipos del CTS.

> Algunos lenguajes .NET, como C#, implementan el CTS utilizando la conveniencia sintáctica del *alias*. Esto quiere decir que si el CTS define un tipo entero de 32 bits bajo la denominación *System.Int32*, lo más probable es que el lenguaje .NET cree el alias «*int*» para dicha clase. Así, en vez declarar una variable como:
>
> System.Int32 x; podría ser declarada de igual forma mediante la expresión int x;

El pilar que resta para hacer posible la interoperabilidad de lenguajes es el **CLS** o **Common Language Specification**.

Aunque se trabaje con código administrado, no hay garantía de que la funcionalidad de los tipos creados por el usuario en un lenguaje puedan ser completamente aprovechados por otros lenguajes. Esto se debe principalmente a que cada compilador usa el sistema de tipos y metadatos para soportar su propio conjunto de características del lenguaje. Si el desarrollador no sabe en qué lenguaje estará el código cliente, no tiene forma de saber de antemano cuáles características serán accesibles para dicho código cliente.

Para que un componente pueda interactuar de forma completa con otro sin importar los lenguajes de implementación, debe exponer únicamente las características comunes a todos los lenguajes intervinientes. El CLS es el conjunto de características básicas de los lenguajes que deben soportar todos los compiladores .NET para generar código compatible.. Las reglas del CLS definen un subconjunto del CTS: todas las reglas que se aplican al CTS se aplican al CLS, excepto cuando el CLS define reglas más estrictas.

Se basa en el principio que MSIL es un lenguaje cuya especificación incluye numerosas características, y es posible que algunos diseñadores de compiladores prefieran restringir las capacidades de su compilador para soportar sólo un subconjunto de lo que ofrece MSIL y el CTS. Esto es válido, con la condición que el compilador soporte cada regla que establece la especificación común de lenguajes.

Hay una serie de requerimientos para que un compilador genere código compatible CLS:

- todos los tipos que aparecen en el prototipo de un método deben ser compatibles con la CLS
- los elementos de un array deben tener elementos de tipo compatible con la CLS

- una clase compatible debe heredarse de otra clase compatible. *System.Object* es compatible con CLS
- dado que los nombres de métodos de una clase compatible CLS no son sensibles a mayúsculas/minúsculas, no pueden existir dos nombres de métodos que difieran solamente en mayúsculas/minúsculas
- las enumeraciones solo pueden ser de tipos *Int16, Int32 o Int64*

Es importante observar que todos estos requerimientos son únicamente para miembros *públicos*[42] y *protegidos*. Los miembros *privados*, dado que no son expuestos, pueden no ser compatibles con la CLS y aun así la assembly será compatible.

¿Qué lenguajes pueden ser interoperables en el Compact Framework?

Teóricamente, pueden ser interoperables todos los lenguajes .NET, es decir, aquellos cuyos compiladores generan código MSIL compatible con el CTS y la CLS. En la práctica, VisualStudio .NET 2003 sólo soporta dos lenguajes para el desarrollo de aplicaciones móviles: VB .NET y C#. Si bien este entorno de desarrollo no es condición necesaria para generar aplicaciones móviles .NET, considero que su utilización es recomendable por las facilidades que ofrece para administrar las diferencias de formato que poseen los dispositivos que correrán las aplicaciones.

Desarrollo para varias plataformas

Actualmente, las plataformas de software que soportan el Compact Framework son:

- Pocket PC 2000
- Pocket PC 2002
- Pocket PC 2002 Phone Edition
- Pocket PC 2003
- Windows CE .NET a partir de la versión 4.1
- Smartphone 2003

El desarrollo de aplicaciones móviles haciendo uso del Compact Framework sigue los mismos principios generales de portabilidad que el .NET Framework estándar. Esto es, cualquier plataforma de software que tenga una implementación del Compact Framework podría correr una aplicación .NET.

No obstante, cuando se desarrolla software para equipos portátiles nos encontramos con varios aspectos de consideración que condicionan la portabilidad directa de las aplicaciones:

- los diferentes tamaños de pantalla de los dispositivos móviles
- las distintos mecanismos de entrada de datos
- las distintas formas y capacidades de almacenamiento permanente
- las distintas capacidades de procesamiento

Esto no significa que no sea posible la portabilidad entre plataformas. Sí indica que las aplicaciones requieren importantes adaptaciones en el diseño y codificación.

No es igual el área de visión de un equipo con Pocket PC que expone su interfaz en una superficie de 240 x 320, que el área de visión de un teléfono con SmartPhone cuyas áreas posibles son de 176 x 220, 160 x 240 y 208 x 240. Por otro lado, la entrada de datos en Pocket PC es mediante un pequeño teclado emulado por software, en cambio en Smartphone la entrada es por medio de las teclas alfanuméricas del equipo hardware. Estas diferencias determinan que las interfaces gráficas de usuarios sean altamente dependientes del dispositivo donde se ejecutarán, por tanto la portabilidad deja de ser una cuestión asociada netamente al Framework y pasa a ser compartida entre el hardware y el Framework.

A los fines de analizar las opciones de portabilidad, en esta tesina se definen tres grandes grupos de plataformas: Pocket PC versión x, Windows CE .NET y SmartPhone 2003. A continuación se caracterizan brevemente los tres grupos y en la tabla 2.3 se muestra una síntesis de ellos.

Pocket PC[43]

Dentro de este grupo se incluyen las distintas versiones de Pocket PC, que han sido construidas basándose en implementaciones sucesivas de Windows CE. Incorporan nueva funcionalidad, interfaces de usuario y aplicaciones optimizadas para dispositivos móviles Pocket PC. Un dispositivo Pocket PC es un PDA basado en Windows CE que incluye aplicaciones personalizadas para cada manufacturera de hardware que las comercializa. La estandarización de los requerimientos de hardware y de software le ha permitido el soporte al desarrollo de aplicaciones por terceras partes.

42. La última versión 2003, se encuentra bajo la denominación original Windows Mobile e incluye a Smartphone 2003
43. En inglés Software Development Kit, kit de software para desarrolladores.

Windows CE .NET

Está diseñado para funcionar en un conjunto mucho más amplio de dispositivos empotrados, portátiles o no. Debido a la gran cantidad de dispositivos sobre los que tiene potencial aplicación no es posible estandarizar los requerimientos de software y de hardware. No hay indicaciones acerca del tamaño de la pantalla, ni siquiera si debe tener pantalla, los mecanismos de entrada / salida, etc.

Este sistema operativo puede ser armado por componentes a elección. Las manufactureras de dispositivos pueden tomar los componentes que necesitan y configurar sistemas operativos personalizados para una amplia gama de necesidades.

SmartPhone 2003

Fue desarrollado para correr en Smartphones. Un Smartphone es un teléfono celular inteligente cuya implementación se basa en Windows CE 3.0.

Tabla 2.5 Síntesis de los grupos de plataformas

	Dispositivos Objetivo	Requerimientos estándar de HW y SW?
Pocket PC	dispositivos móviles tipo PDAs	si
Windows CE .NET	dispositivos con software empotrado	no
Smartphone 2003	teléfonos celulares Smartphones	si

La definición de requerimientos estándar de hardware y software tiene un impacto importante a la hora de portar aplicaciones entre plataformas. Los requerimientos de hardware nos indican qué características debe poseer el hardware sobre el que correrá la plataforma de software. Si dichos requerimientos son estándar es más factible desarrollar aplicaciones portables porque conocemos de antemano las opciones de configuración del hardware.

De acuerdo a lo expuesto podemos formular ahora el siguiente principio:

Cada uno de estos tres grupos tiene características únicas para el desarrollo de aplicaciones.

Como ejemplo, el SDK[44] de Smartphone 2003 [MS03.2] señala algunas diferencias entre el desarrollo para Pocket PC y para Smartphone bajo el Compact Framework:

- Smartphone posee solo un subconjunto de los controles de Pocket PC
- La interacción en Smartphone es por el teclado alfanumérico, no existe pantalla sensitiva.
- La memoria RAM de Smartphone se borra cada vez que se apaga el teléfono. Para guardar los datos en la memoria permanente deben direccionarse los datos a la memoria flash permanente.
- Las restricciones se memoria son aún mayores en Smartphone y algunas aplicaciones pueden no llegar a ejecutarse.
- El sistema de archivos es muy pequeño y debe ser usado con moderación. Si se necesitan guardar datos extensos deben usarse tarjetas de memoria auxiliares.

Conclusiones

La portabilidad en el Compact Framework es una función de dos variables: las plataformas en cuestión y los recursos que utiliza la aplicación.

Todas las diferencias de las plataformas, físicas en su mayoría, determinan que la codificación de aplicaciones portables no sea un proceso automático. Demandan que haya variaciones en el comportamiento de muchas clases en el .NET Compact Framework. Estas variaciones les permiten a los desarrolladores construir aplicaciones para cada plataforma sin la necesidad de aprender completamente nuevos entornos para cada una [MS03.3].

Los recursos que utilizará la aplicación deben estar presentes de la misma forma en las plataformas objetivo. Si algún recurso no se encuentra en una de ellas, es necesaria una adaptación del código. Un ejemplo es el almacenamiento permanente en memoria flash en Smartphones y en memoria RAM en Pocket PC.

En algunos casos pueden utilizarse técnicas de doble codificación, es decir, se evalúan condiciones para saber sobre qué plataforma se está ejecutando y en base a eso activar una sección de código u otra. No obstante, este último enfoque tiende a extender considerablemente la longitud del código y a escribir

44. En referencia a las técnicas de programación de aplicaciones estándar de Windows de escritorio.

varias versiones de lo mismo, lo cual no parece estar acorde a las prácticas convencionales de escritura de código portable.

A continuación se expone una tabla indicando cuándo se requieren mayores adaptaciones para un lograr código portable entre pares de plataformas.

Tabla 2.6. Portabilidad directa entre plataformas

	Pocket PC 2000	Pocket PC 2002	Pocket PC 2002 PE	Pocket PC 2003	Windows CE .NET	SmartPhone 2003
Pocket PC 2000		✓	✓	✓	🛠	🛠
Pocket PC 2002			✓	✓	🛠	🛠
Pocket PC 2002 PE				✓	🛠	🛠
Pocket PC 2003					🛠	🛠
Windows CE .NET						🛠

La portabilidad es más directa


Puede requerir adaptaciones mayores

3.1 Indicaciones para el desarrollo eficiente en dispositivos móviles

Gestión de los recursos limitados

En el capítulo 1 se expusieron las limitaciones generales más frecuentes que poseen los dispositivos móviles actuales. Muchas de estas limitaciones determinan que deban contemplarse aspectos adicionales en el proceso de desarrollo de las aplicaciones móviles. Un ejemplo es el diseño de las interfaces gráficas de usuario, que deben ser adaptadas a los tamaños de pantalla disponibles y a los distintos mecanismos de entrada/salida ofrecidos por las plataformas; temática que será cubierta en la siguiente sección de este mismo capítulo.

No obstante, existen más aspectos que requieren un cambio sustancial en la forma de diseñar e implementar las piezas de software, que necesitan ser adecuadas a importantes restricciones de capacidad de procesamiento y de almacenamiento temporal y permanente. Implica que deben modificarse algunas prácticas *habituales*[44] en el uso de los recursos, principalmente de memoria y procesamiento. El uso de cada estructura de datos y métodos a implementar debe ser planificado, teniendo en cuenta la necesidad de uso y el impacto que tendrá en el rendimiento del sistema. Por ello se plantean a continuación una serie de técnicas y recomendaciones para minimizar el uso de recursos escasos, y de avisos a considerar sobre el funcionamiento de algunos componentes del Compact Framework que pueden afectar a la *performance*, concepto que definiré a continuación.

***¿Qué es la performance?* [MS03.1]**

La performance es una medida cuantitativa del grado de respuesta que tienen las aplicaciones en relación al usuario. El grado de respuesta tiene que ver con el tiempo que le lleva a una aplicación responder a una entrada de usuario, aunque el sistema esté ocupado realizando otras tareas. Existen dos principios que afectan la performance de una aplicación:

1. la performance absoluta: es la que corresponde al sistema (poder de procesamiento)
2. la performance aparente: corresponde a la aplicación en base a cómo fue programada

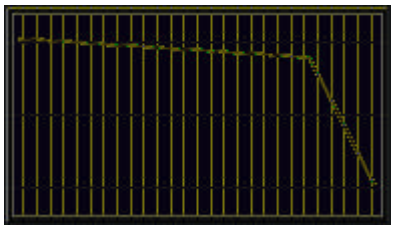
¿Qué relación tiene el uso de memoria y la performance en el Compact Framework?

A medida que las aplicaciones van solicitando más memoria al gestor de ejecución, la performance se reduce gradualmente con una pendiente poco pronunciada. Al llegar a un nivel aproximado del 80% de utilización, el entorno debe ejecutar una serie de acciones (como *code pitching*[45]) para obtener más memoria disponible y se reduce la performance de manera abrupta.

La gráfica 3.1 muestra este comportamiento del Compact Framework:

45. Algunos métodos ya compilados a código nativo por el compilador JIT son removidos de la memoria caché.

Figura 3.1. Performance del Compact Framework



Nota Importante

Si se desarrollan aplicaciones para que funcionen tanto en Pocket PC como en Smartphone, deben diseñarse de acuerdo a las restricciones del mínimo común denominador que es Smartphone. Sabemos del capítulo 1 que estos dispositivos son mucho más restringidos en procesamiento y memoria que los Pocket PC.

Garbage Collection[46]

El CLR posee un mecanismo de administración de la memoria denominado «recolector de basura» que permite mantener bajo su control de todos los objetos instanciados en memoria.

En condiciones de poca memoria disponible, este mecanismo es activado por el gestor de ejecución para recuperar memoria no utilizada efectivamente por las aplicaciones. Durante la recolección se examina el código para evaluar si existen referencias a objetos creados en el *heap*. Aquellos objetos no referenciados serán eliminados y el gestor devuelve el espacio de memoria para reuso posterior. Los objetos que aún son referenciados se reacomodan en memoria para maximizar la contigüidad. Este reordenamiento dinámico tiene un impacto en la performance en las aplicaciones que se ejecutan al momento de la recolección: todos los *threads* necesitan ser suspendidos momentáneamente para poder reacomodar segmentos de datos y código. Suponiendo que tras la activación del recolector no se obtiene la cantidad de memoria deseada, el código perteneciente a un método alojado en memoria caché puede ser removido para tomar el espacio. Si el método es invocado nuevamente, debe ser recompilado.

Tal como se explicó en el capítulo 2, el recolector es no determinístico y por eso no puede ser anticipada su activación. No obstante, la clase *GC*[47] del Compact Framework ofrece un servicio de activación determinística del recolector a través del método *GC.Collect*. Puede utilizarse este método inmediatamente que se han dejado de referenciar una gran cantidad de datos y se considera oportuno liberar espacio en memoria. Normalmente, no debería utilizarse este servicio debido a que consume recursos de procesamiento y puede llevar a una importante disminución de la performance mientras se ejecuta. No se recomienda en absoluto su uso en aplicaciones móviles.

Destructores, finalizadores[48] e interfaces[49] de finalización

Los constructores son métodos para la creación de objetos en memoria en estado de consistencia, es decir, no sólo asignan una porción de memoria al objeto sino que, además, el objeto es inicializado y dejado en condiciones apropiadas para su funcionamiento.

Los **destructores** son el concepto contrario de los constructores, su función es terminar el ciclo de vida de un objeto liberando todos los recursos utilizados: finalizando procesos abiertos, cerrando conexiones establecidas, devolviendo memoria solicitada, etc. Es particularmente importante para aplicaciones de dispositivos móviles liberar los recursos lo antes posible[50].

No es necesario proveer una implementación para un destructor, el gestor de ejecución se encarga de ello a través del mecanismo de recolección recientemente explicado. En la mayoría de los casos esto es suficiente. Sin embargo, si se utilizan recursos del sistema operativo como archivos compartidos o conexiones, es necesario tomar acciones explícitas para asegurar que tales recursos sean liberados: solo para el caso de recursos que no se encuentran bajo el control del CLR. Los recursos administrados se liberan automáticamente. Los destructores en .NET se implementan por medio de **finalizadores**. Son un mecanismo provisto por .NET para tratar de asegurar la liberación de recursos utilizados por un objeto fuera de lo que comprende la imagen de memoria (atributos y métodos) del objeto en sí, como se dijo recientemente.

46. Concepto que se introdujo en el capítulo 2.
47. La clase GC es acrónimo de Garbage Collection y representa la funcionalidad necesaria de este mecanismo.
48. De la terminología .NET
finalizers
49. Las interfaces son una característica de los lenguajes orientados a objetos que permiten declarar que una clase implementa una funcionalidad determinada.
50. Especialmente los recursos de memoria, que son muy escasos en algunos equipos

Se proveen implementando el método *Finalize( )*[51] en una clase. Los finalizadores deben ser utilizados con precaución dado que tienen algunas implicaciones:

- La finalización ocurre en otro thread de ejecución y resta eficiencia.
- Los objetos con finalizadores permanecen por más tiempo en el sistema y consecuentemente introducen una sobrecarga: se requieren al menos dos recolecciones automáticas hasta que son liberados. Esto se debe a que un objeto que contiene un finalizador es incorporado a una lista de finalización. Cuando el recolector termina sus tareas inicia los finalizadores de los objetos que están en la lista de finalización y los remueve de tal lista. En la próxima recolección, los objetos son liberados de memoria.
- No hay garantía del orden de finalización: si un objeto A contiene una referencia a un objeto B, éste último no estará disponible si A lo intenta utilizar durante una finalización.

.NET provee una **interfaz de finalización** también disponible para el Compact Framework denominada *IDisposable*. Los usuarios de una clase que implementan la interfaz deben proporcionar un método público[52] llamado *Dispose* que será llamado directamente para liberar los recursos no administrados por el gestor de ejecución. Si se implementa este método, *Finalize* se convierte en una medida de seguridad para el caso en que *Dispose* no sea invocado. La correcta implementación del método *Dispose* provoca que *Finalize* no sea invocado, con la consecuente ventaja de performance.

Stack y Heap[53]

Esta sección explica brevemente las principales diferencias de las dos formas principales de alojamiento en objetos en memoria que ofrece .NET: memoria dinámica en el heap y memoria estática en el stack.

El término **memoria dinámica** se refiere a la petición de memoria en tiempo de ejecución que una aplicación hace al CLR. Una aplicación puede solicitar porciones de memoria tantas veces lo requiera siempre y cuando el CLR cuente con memoria disponible para entregarle. La memoria dinámica se solicita típicamente al instanciar un objeto. El lugar donde se ubican tales objetos es el montón, una región administrada por el CLR y por ende de «jurisdicción» del recolector de basura. Esta forma de administración de memoria tiene la flexibilidad de ocupar sólo el espacio requerido, no es necesario hacer reservaciones anticipadas de memoria sino que es posible solicitarla dinámicamente a medida que se necesita. También tiene la ventaja de ser liberada en tiempo de ejecución cuando ya no es utilizada. Sin embargo, existe una penalización en la performance del sistema para los objetos que se manejan en memoria dinámica, pues existen referencias[54] a tales objetos y para accederlos se requieren operaciones que consumen recursos de procesamiento. A causa de las referencias, estos objetos se denominan en .NET como «tipos de datos por referencia».

Para aplicaciones móviles que emplean estructuras de datos de numerosos elementos en memoria, el uso de objetos dinámicos puede significar una disminución del rendimiento. En circunstancias como éstas podría evaluarse el uso de la memoria estática e introducir consecuentemente una mejora en la eficiencia.

Por otro lado, el término de **memoria estática** se refiere a la memoria que es asignada a una unidad de programa[55] por única vez y durante su creación. Tal porción de memoria permanece ocupada durante toda la vida de la unidad de programa y no puede ser liberada en tiempo de ejecución explícitamente.

Supongamos que, a medida que una aplicación se ejecuta, un puntero[56] va apuntando secuencialmente a la próxima instrucción a ejecutarse. Si la instrucción hace referencia a un objeto estático que se halla fuera del ámbito de la unidad de programa, éste no estará disponible para su uso. El uso de memoria dinámica ofrece más flexibilidad en cuanto al manejo de ámbitos, pero la ventaja de la asignación en el stack es la eficiencia que ofrece: posee un esquema de administración mucho más eficiente que el heap. Tales objetos se denominan en .NET «tipos de datos por valor» y se debe a que no requieren referencias para ser accedidos.

A modo de síntesis, la tabla 3.1 a continuación muestra al lector una guía rápida evaluar qué método de asignación se recomienda usar en aplicaciones móviles de acuerdo al volumen de datos:

---

51. Cada lenguaje .NET posee una forma propia para definir el método Finalize( ). En C#, por ejemplo, se realiza precediendo un método de igual sintaxis del constructor con el símbolo ~ y el compilador genera el método Finalize automáticamente
52. Se refiere al modificador de accesibilidad que permite al método ser accedido desde cualquier clase.
53. Algunos textos en español hacen referencia a los términos pila y montón respectivamente.
54. Las referencias son variables descriptoras del objeto, situadas externas al heap, que poseen como atributos el nombre del objeto y su contenido es la dirección de memoria del objeto en el heap.
55. Se entiende por unidad de programa a métodos, módulos o bloques que tienen características propias en cuanto a su ámbito y visibilidad.
56. En un sentido estricto, este puntero podría ser el registro de CPU llamado IP (Instruction Pointer). Sin embargo, no es apropiado para este ejemplo meramente ilustrativo.

Tabla 3.1 Esquemas de asignación de memoria en .NET

Asignación	Ubicación	Tipos	Performance	Volumen de datos deseable
Dinámica	heap	Por referencia	Menos eficiente	Pequeña cantidad
Estática	stack	Por valor	Más eficiente	Grandes cantidades

Uso de Arrays y Tablas

El array es una estructura de datos simple de gran utilidad en la mayoría de las aplicaciones, también conocida como vector. Tiene la particularidad de ser una estructura homogénea, es decir, todos sus elementos son del mismo tipo[57]. La forma de asignación de memoria más común en un ambiente orientado a objetos es de forma dinámica por la flexibilidad que ofrece. No obstante, el uso de arrays estáticos es mucho más eficiente. Se recomienda su uso en las aplicaciones móviles si la cantidad de elementos que posee es numerosa.

Independientemente del método de asignación de memoria, el uso de arrays debe quedar limitado sólo a aquellas situaciones en que sean necesarios, a causa de la escasa cantidad de memoria que cuentan algunos dispositivos. Para saber si la cantidad de bytes que ocupa el array es considerable en relación a la memoria disponible en el dispositivo en particular, puede aplicarse el operador *sizeof* a la clase de un elemento y multiplicar el valor que devuelve el operador por la cantidad de elementos que contiene efectivamente el vector.

Si se emplea el lenguaje C# para implementar la aplicación móvil, puede utilizarse la palabra reservada *stackalloc* para solicitar memoria en la pila. El uso de este operador requiere manejo de código no administrado, lo que podría introducir problemas de debugging[58] si no se hace un correcto manejo de punteros durante la codificación.

El Compact Framework ofrece dos formas básicas para la creación y manejo de arrays dinámicos, mediante las clases *Array* y *ArrayList*. Si bien ambas clases hacen uso de memoria dinámica, se diferencian en la forma que adquieren la memoria del sistema:

- *Array* requiere conocer la capacidad del vector en el momento de su creación, y se mantendrá esa dimensión durante la vida del objeto (hasta que sea recolectado). Se tiene un control sobre la máxima cantidad de bytes que requiere.
- *ArrayList* permite variar dinámicamente la capacidad del vector bajo demanda. Se le asigna una capacidad inicial y si ésta es excedida, se duplica automáticamente. No se tiene un control sobre la cantidad máxima de bytes que esta colección de datos requiere.

El programador debe modificar algunas prácticas habituales a la hora de implementar estructuras de datos en aplicaciones móviles. Para aplicaciones estándar de escritorio, el programador asigna generalmente una capacidad superior a las estructuras de datos en memoria, con el objetivo de mantener un buen margen de funcionamiento. En aplicaciones móviles el tamaño de las estructuras debe estar bien medido y los márgenes deberían ser mínimos. Además, es recomendable prestar atención a los detalles de implementación que puedan reducir los requerimientos de memoria. Vemos esto con un ejemplo ficticio que puede ser ilustrativo.

Ejemplo:

Se tiene una matriz bidimensional a fin de almacenar la computación de distancias entre algunas ciudades principales del país:

Tabla 3.2 Matriz «M» de computación de distancias de ciudades

	Buenos Aires	Rosario	Córdoba	Santa Fe	Viedma
Buenos Aires	0	320	750	470	1200
Rosario	320	0	400	160	1050
Córdoba	750	400	0	200	1150
Santa Fe	470	160	200	0	1100
Viedma	1200	1050	1150	1100	0

Si se observa con atención, se cumplen las siguientes propiedades para la matriz M:

M[i , j] = M[j , i] para todo i ‘« j

M[i , j] = 0 para todo i=j

57. Con esta definición no se excluye el uso del esquema de herencia ni el principio de sustitución de Liskov, que establece que un subtipo puede ocupar el lugar de un tipo base.

58. Proceso de depuración del código, en el cual se intentan corregir errores en la codificación.

Esto implica que en vez de crear en memoria una matriz de 25 posiciones, podríamos crear una de tan solo 10 posiciones. Algunos lenguajes de .NET como C# (que funciona en el Compact Framework), ofrecen facilidades para la implementación de matrices ortogonales. Una representación ortogonal podría emplear las propiedades mencionadas con la siguiente representación en memoria:

Tabla 3.3 Matriz «N» ortogonal de computación de distancias de ciudades

	Rosario	Córdoba	Santa Fe	Viedma
Buenos Aires	320	750	470	1200
Rosario		400	160	1050
Córdoba			200	1150
Santa Fe				1100

Como puede verse, es un ejemplo de matrices pequeñas con el fin de ilustrar la problemática de implementación. Por lo general, en aplicaciones de escritorio no vale el esfuerzo de complejizar el código por ahorrar unos tantos bytes de memoria, pero en dispositivos móviles muy restringidos (como es el caso actual de los Smartphone) puede resultar una buena práctica.

Estos conceptos también se extienden al uso de tablas para representar datos y relaciones. El objeto *DataSet* es la parte principal del manejo de datos en ADO.NET[59]. Un DataSet representa una memoria caché interna, que contiene objetos *DataTable* a su vez formado por objetos *DataRow* y *DataColumn*. Es decir, un *DataSet* es un contenedor offline[60] de información proveniente de alguna fuente de datos, y se lo estructura mediante objetos que proveen la abstracción necesaria para el manejo de tablas, filas y columnas respectivamente. Estos objetos son de uso muy frecuente cuando se necesita manejo y acceso a datos, inclusive en el ámbito de las aplicaciones móviles. Por ello deben tenerse las mismas consideraciones de utilización de espacio de memoria que el caso de los arrays. Por ejemplo, si se está representando una base de datos con agendas donde cada registro puede tener hasta 5 direcciones de correo, es deseable que no demanden los 5 lugares en memoria aquellos registros que contienen solo una dirección de correo.

Bases de datos: SQLServer CE[61] y acceso remoto a datos

Es frecuente que las aplicaciones necesiten contar con algún repositorio de datos. Cuando se precisan almacenar datos con un nivel de administración más avanzado que el simple uso de tablas u hojas de cálculo, se acude a herramientas como los administradores de bases de datos relacionales. Estos administradores ofrecen un soporte integral para el manejo de los datos, que va desde su almacenamiento en formatos adecuados, funciones de manipulación, consulta, actualización, transacciones, hasta el manejo de la seguridad.

En el ámbito de las aplicaciones móviles, un ejemplo típico que evidencia la necesidad de administradores de bases de datos es el de vendedores viajantes. Los vendedores se trasladan hacia los lugares donde se hallan sus clientes y pueden llevar a cabo operaciones comerciales de consultas de catálogos, emisión de presupuestos, reservas y ventas. A medida que avanzamos en el razonamiento del ejemplo surgen algunas cuestiones como:

- ¿puede un dispositivo móvil alojar en su memoria un administrador de bases de datos?
- ¿puede una aplicación móvil acceder a una base de datos remota?
- ¿puede una aplicación móvil actualizar sus datos locales con una base remota?

En el capítulo 2 vimos que el Compact Framework incluye, como parte de su librería de clases, un subconjunto del Framework estándar y un superconjunto desarrollado especialmente par el uso en aplicaciones móviles. En tal superconjunto existe un grupo de clases bajo el espacio de nombres *System.Data.SqlServerCE*. Contiene clases que proveen la funcionalidad necesaria para tener acceso a datos de un servidor SQLServer CE en un entorno administrado.

En la lista oficial [MS03.4] de los requerimientos de software y hardware de SQLServer CE se indica que se necesitan entre 1 y 3 MB de espacio de almacenamiento para un dispositivo cliente, dependiendo de los componentes instalados y del procesador que posea el equipo. La gama de equipos portátiles es amplia y

59. ADO.NET proporciona acceso a fuentes datos tales como servidores de bases de datos. Es el medio para conectarse, acceder, manipular y actualizar fuentes de datos desde una aplicación .NET.

60. En inglés, significa «fuera de línea», que en este caso indica que no forma parte de una base de datos real sino que es una representación interna para uso de una aplicación particular .NET

61. SQLServer CE es un producto de software de la compañía Microsoft, que implementa un administrador reducido de bases de datos local pensado para un desarrollo rápido de aplicaciones que extienden la funcionalidad empresarial mediante el uso de dispositivos móviles. Requiere como sistema de base Windows CE.

como ya hemos visto, las capacidades de memoria y almacenamiento varían de uno a otro. No obstante, podemos responder la primera interrogante de manera afirmativa: seleccionando adecuadamente los componentes, es posible instalar un administrador de bases de datos en un PDA, XDA o cualquier dispositivo que cumpla los requerimientos de memoria y procesamiento.

Una decisión de diseño distinta puede ser tener acceso remoto a una base de datos que no se halla instalada localmente como es el caso de SQLServer CE. La aplicación podría establecer una conexión a una base de datos distante y obtener el acceso a datos necesario para operar, sin consumir el espacio de memoria local. Sin embargo sabemos que, por las características del contexto móvil, no puede garantizarse siempre una conexión a un equipo remoto. Hay condiciones de interferencia, lejanía, climáticas, etc., que pueden provocar la imposibilidad de establecer la conexión a otros equipos. Por estas causas, una aplicación móvil que delega el almacenamiento de un administrador de bases de datos debe estar preparada para conectarse remotamente, continuar su operación con funcionalidad reducida si la conexión se interrumpe abruptamente y ser capaz de reconectarse para retomar la actividad. Otro factor de consideración para las conexiones es el costo y el estado de madurez actual de las comunicaciones inalámbricas.

En [MS03.1] se indicó irónicamente que «hoy en día las comunicaciones son un privilegio para las aplicaciones móviles, no un regalo de cumpleaños». Esto puede interpretarse en el sentido que deben emplearse sólo en los momentos de necesidad y estar preparados para prescindir de ellas cuando no estén disponibles, como se dijo en el párrafo anterior.

El Compact Framework, en respuesta a la segunda interrogante, provee mecanismos para el acceso remoto a datos a través de una lista de clases. Algunas clases de hallan en el espacio de nombre *System.Data.SqlClient* (cliente de SQL). Este espacio contiene clases para establecer la conexión, obtención de permisos de acceso, manejo de condiciones de error o no disponibilidad, realización de consultas, actualizaciones, transacciones, etc. En la sección previa de «*Uso Arrays y Tablas*» se expusieron clases que permiten el modelado de contenedores de información fuera de línea (offline). Estos contenedores, también llamados *caché*, pueden emplearse para alojar la información recuperada a través de una conexión remota. El proceso de acceso a datos implica que una cantidad de información es conducida al dispositivo móvil. La aplicación puede usar la información para realizar consultas simplemente, o puede introducir cambios que necesitarán ser actualizados en el servidor central. Durante los períodos de desconexión, se accede a los datos del *caché*.

Si solamente se accede a la base para realizar consultas, el proceso sería de la siguiente forma:

1. establecer conexión
2. recuperar datos
3. cerrar conexión

} Período rápido (podrían ser minutos)

Para las consultas seguidas de actualizaciones que deben reflejarse en el servidor remoto:

1. establecer conexión
2. recuperar datos
3. cerrar conexión

} Período rápido (podrían ser minutos)

4. actualizar datos localmente[62] } Período indeterminado
5. establecer conexión
6. enviar actualizaciones al servidor
7. cerrar conexión

} Período rápido (podrían ser minutos)

El esquema anterior puede interpretarse mediante un ejemplo: un proveedor viajante cuenta con un dispositivo móvil para realizar transacciones de venta. El equipo no cuenta con una base de datos local por cuestiones de capacidad de almacenamiento. En cambio, el proveedor se conecta cada mañana al servidor central remoto, baja un subconjunto del catálogo de ventas de acuerdo a la zona en que se encuentre y se desconecta (pasos 1,2 y 3). Los datos obtenidos se almacenan localmente en un contenedor offline de información. Durante el día realiza transacciones de venta sobre el dispositivo (paso 4) y al finalizar la jornada tiene que enviar las transacciones al servidor central (pasos 5,6 y 7). De esta forma, el proveedor tuvo acceso a información del sistema de operaciones de la empresa sin necesidad de «hospedar» una extensa base de datos localmente.

Para responder la última pregunta se enuncia lo siguiente. La frecuencia con que una aplicación móvil debe actualizar la información local/remota está determinada por las reglas del negocio. En aplicaciones de

62. Utilizando la clase DataSet u otros contenedores de información offline (no necesita mantener la conexión con el origen de los datos).

venta podría bastar una actualización del catálogo una sola vez a la semana, en otras podría ser mucho más frecuente. Las actualizaciones no tan frecuentes o aquellas muy voluminosas pueden ser realizadas en el mismo ámbito de la empresa, sin necesidad de conexiones remotas. En cambio, si existe la necesidad de actualización a distancia, el Compact Framework provee mecanismos para ello. Para actualizar datos se emplean mecanismos de sincronización. Entendemos por **sincronización** al proceso que deja a dos o más contenedores de datos en un estado de convergencia, es decir, se realizan acciones para que sus contenidos sean homogéneos. Tales acciones implican que una aplicación envíe los cambios en su contenedor local al servidor central, y que la aplicación reciba los cambios que otros hayan realizados sobre la base central. Si la aplicación solamente requiere obtener datos remotos, no hablamos de sincronización sino de **acceso a datos**.

La tabla 3.4 a continuación establece una comparación para el acceso a datos local y remoto. Local implica que existe un administrador de bases de datos SQLServer CE en el dispositivo. Remoto implica que se accede a un servidor central no local.

Tabla 3.4 Comparación del acceso a datos local y remoto

Local	**Remoto**	
Disponibilidad	siempre	temporal
Conexión a base remota	no requerida	requerida
Espacio de memoria local	considerable	mínimo
Eficiencia	buena	reducida
Sincronización	requerida	requerida si se dispone de un caché en el dispositivo

Criptografía

El Compact Framework actualmente no provee clases en su librería FCL para encriptar datos ni conexiones. No obstante, Windows CE contiene una API que da soporte para la criptografía y Pocket PC incluye una API denominada CAPI (CryptoAPI). En tales interfaces de programación se incluyen implementaciones de algoritmos de seguridad. Se recomienda evaluar qué algoritmos emplear y las longitudes de las claves, dado que a medida que la longitud de la clave aumenta se requieren más recursos de procesamiento y la performance se reduce notablemente. Para el algoritmo RSA se recomienda no exceder los 512 bits en las claves [SDK03.1].

Manejo de Threads

Multithreading es la capacidad de un sistema operativo o entorno de ejecución de trabajar con mútiples threads. En el capítulo 2 he expuesto que un *thread* es una unidad de procesamiento que en general requiere menos recursos que un proceso tradicional. El Compact Framework permite desarrollar aplicaciones aprovechando las características de multithreading y provee una serie de clases de utilidad que abstraen el manejo de esta facilidad. Las clases para el manejo de threads se encuentran organizadas en el espacio de nombres *System.Threading*.

Con el debido uso de threads, una aplicación puede lograr:

- tener más **capacidad de respuesta** (ver definición de performance en este capítulo)
- ser más **escalable**, es decir, que el diseñador puede incorporar nuevos requerimientos de manera relativamente sencilla, sin necesidad de rediseñar la aplicación.
- Algunas aplicaciones pueden tener mejor **productividad**: si una aplicación puede ser divididas en partes separadas que se procesan en paralelo[63], el procesamiento global puede lograrse más rápidamente.

Esta característica puede ser empleada para mejorar la eficiencia en los dispositivos móviles con capacidades restringidas de procesamiento.

En el párrafo anterior he remarcado la palabra «debido» refiriéndome al uso de los threads. La razón de la aclaración es que multithreading es una característica de programación avanzada y requiere tener precauciones a la hora de implementarla. Su uso debe ser limitado a aquellos casos en que las aplicaciones tienen períodos de inactividad, de manera de poder asignar el procesador a otras tareas y así incrementar la performance del sistema. Además, el uso de threads introduce algunos problemas potenciales que deben ser gestionados por el implementador:

63. En un sentido estricto, el procesamiento paralelo ocurre únicamente en equipos que poseen más de un procesador. Sin embargo, el modelo de procesamiento multithreaded ofrece un pseudo paralelismo que es capaz de compartir el uso de un único procesador entre varios threads a intervalos de tiempo despreciables. Esto da al usuario la idea de procesamiento paralelo.

- un thread puede interferir con los datos de otro thread
- si se crean muchos threads la performance puede reducirse
- pueden presentarse problema de comunicación y sincronización entre threads.

Estos temas requieren un tratamiento exhaustivo aparte y quedan fuera del alcance de esta tesina.

Autonomía de Batería en Smartphone

Basémonos en el principio que la ejecución de código consume baterías. Como se indicó en el capítulo 1, el modo de administración de la energía en Smartphone es bajo el esquema de «modo *idle»,* en el cual el teléfono necesita estar siempre encendido para poder recibir llamadas. Si no hay ejecución de código, el equipo entra en modo de bajo consumo. No obstante, esto no ocurre automáticamente: las aplicaciones deben tener algunas consideraciones de diseño para que pueda activarse efectivamente el modo de ahorro:

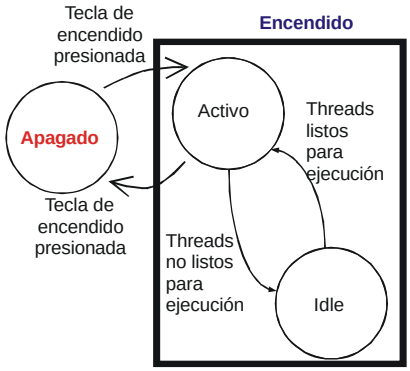
- las aplicaciones deben efectuar sus procesamientos de manera rápida
- las aplicaciones deben ser construidas de manera que no contengan componentes que se ejecuten frecuentemente: para que el equipo pueda ingresar al modo de ahorro debe tener una inactividad de al menos un segundo. Si la aplicación requiere varios procesamientos por minuto, el equipo no podrá activar las opciones de energía.

Para dar una noción de consumo, el equipo Smartphone en modo activo consume aproximadamente 400mW, mientras que en modo *idle* consume 30mW. Para reducir el consumo es recomendable:

- evitar temporizadores en las aplicaciones
- no emplear animaciones, pues requieren procesamiento constante de código
- si la aplicación no está siendo utilizada[64], el uso de CPU debe ser CERO
- diseñar las aplicaciones en base a un modelo orientado a la interrupción: hay procesamiento únicamente cuando hay interacción con el usuario

El modelo de administración de la energía en Smartphone puede representarse de manera simplificada mediante la siguiente figura tomada de [MS03.1]:

Figura 3.2 Diagrama de estados de administración de energía en Smartphone



Cuando el equipo se encuentra en estado apagado, todos los suministros de energías están deshabilitados. Al presionar el botón de encendido, el equipo pasa al estado de actividad donde el CPU funciona al 100% y las memorias RAM/Flash se encuentran encendidas; este estado es el que consume la máxima cantidad de energía de las baterías. Si el sistema detecta un período en que las aplicaciones o el usuario no requieren procesamiento, el equipo entra en modo de inactividad (idle) donde ocurre un menor consumo de energía que en estado de actividad y no se realiza procesamiento. En este estado la pantalla se mantiene encendida, y el CPU permanece ocioso, es decir, no procesa instrucciones pero está prendido. Una aplicación o interacción del usuario puede provocar el paso de la inactividad a la actividad nuevamente.

Como conclusión, es fundamental diseñar las aplicaciones Smartphone de forma que aprovechen las opciones de ahorro de energía para poder maximizar el tiempo de uso efectivo de los dispositivos en el contexto móvil. Para ello, las aplicaciones deben estar el máximo tiempo posible en modo de inactividad sin requerir procesamiento. El uso de threads con Compact Framwork puede ayudar a lograrlo.

Invocación a Métodos

De manera contraria a lo que indican las buenas prácticas[65] de diseño y codificación, en algunas aplicaciones móviles es preferible extender la longitud de código de un método en vez de contar con varios

64. Excepto aquellas aplicaciones que emplean redes de datos y necesitan procesamiento por períodos determinados.
65. Se desea que la cohesión sea máxima, es decir, que un método realice una única acción.

métodos que requieran múltiples invocaciones. Esto se debe a que cada método invocado requiere una entrada en la pila de ejecución y si la pila se extiende demasiado la performance comienza a comprometerse. Cabe aclarar que la agrupación de funcionalidad en métodos únicos debe hacerse cuidadosamente, es decir, vinculando solamente funcionalidades relacionadas a fin de contar con un código legible y mantenible.

Una extensión de esto es el uso de métodos recursivos, los cuales deben ser evitados.

3.2 Diseño de la interfaz gráfica de usuario

El diseño de interfaces gráficas de usuario es un área de estudio especial para todos los sistemas computacionales y electrónicos. Esto se debe a que son el nexo entre el sistema y el usuario, que en definitiva es quien operará el sistema y para el cual fue creado.

Dando por hecho que un sistema ha sido diseñado y desarrollado atento a la reglas de calidad y a las buenas prácticas, las interfaces de usuario pueden ser elementos determinantes del éxito o fracaso de muchos sistemas. Una interfaz diseñada inadecuadamente, en el mejor de los casos, hará que el sistema sea difícil de utilizar y provocará que los usuarios se equivoquen. En el peor de los casos el resultado puede ser catastrófico, yendo desde el riesgo de vida de sus usuarios hasta el rechazo de utilización de los mismos por la falta de comprensión.

Los sistemas de software, por su naturaleza abstracta, no tienen una forma física. El contacto de los usuarios con el software es a través de las interfaces de usuario, que intentan crear una representación adecuada de la funcionalidad del software para que las personas puedan utilizarlo. Es por eso que los diseñadores de interfaces deben tomar en cuenta las capacidades físicas y mentales de las personas que utilizarán el software. Es propio de las personas cometer errores, olvidos, distraerse. Estos puntos requieren ser considerados a la hora del diseño de interfaces.

En [SOM02] se proponen algunos principios para el diseño de interfaces de usuarios, que por su carácter general pueden ser considerados como un subconjunto de los requerimientos para el diseño en dispositivos móviles. La tabla a continuación muestra estos principios:

Tabla 3.5 Principios generales de diseño de las interfaces de usuario

Principio	Significado
Consistencia	Las operaciones similares o comparables deben invocarse de la misma forma siempre que sea posible. El usuario debe aprender una vez y aplicarlo donde pueda.
Familiaridad de uso	La interfaz debe representar conceptos que son tomados del dominio de sus usuarios más frecuentes. Debe ser lo más simple posible.
Predictibilidad	El comportamiento del sistema no debe provocar sorpresas o acciones inesperadas a sus usuarios.
Capacidad de recuperación	Antes eventualidades o condiciones de excepción, la interfaz debe ofrecer a los usuarios mecanismos de recuperación.

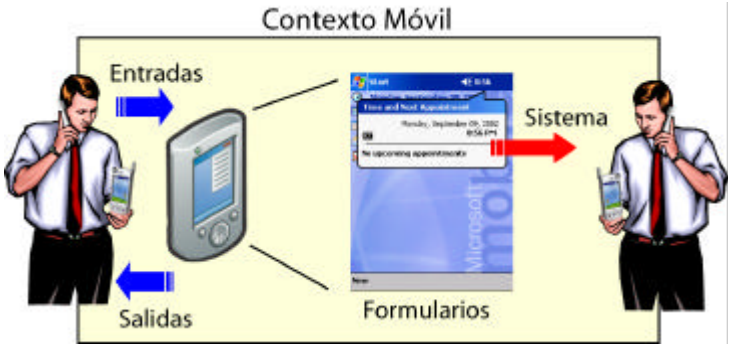
En la conferencia para desarrolladores de aplicaciones móviles de Microsoft celebrada el año 2003 [MS03.1] se amplían estos principios generales para abarcar cuestiones más específicas relacionadas a equipos móviles acorde a sus limitaciones[66]. Los principios ampliados son:

- No intentar siempre que el software móvil sea un reflejo de las aplicaciones de Windows de escritorio. No se puede copiar una interfaz de usuario completa a una interfaz móvil.
- Evitar las opciones. La funcionalidad ofrecida debe ser lo más concreta posible
- Dar más énfasis a los datos y al contenido que a la interfaz gráfica en sí.
- Minimizar el movimiento de los dedos pulgares para Smartphones y minimizar el intercambio repetido del uso de teclado/pantalla. Ejemplo: una aplicación para el ingreso de datos no debe requerir continuamente que el usuario deje de usar la pantalla sensitiva para usar el teclado (físico o emulado por software) o viceversa.
- Considerar que no siempre es posible que un único formulario tenga todo lo que el diseñador desea. Deben tomarse decisiones de eliminar características secundarias o restringir la funcionalidad enfocándose en las tareas claves.

Habiendo planteado los principios podemos comenzar con una guía para el diseño de interfaces en dispositivos móviles haciendo uso exclusivamente del Compact Framework. Para ello plantearé un modelo de 5 componentes básicas: el contexto móvil, los formularios, las entradas, las salidas y la comunicación del usuario con el sistema. La figura 3.3 a continuación muestra el modelo.

66. En el Capítulo 1 sección 1.4 se han expuesto las restricciones de estos dispositivos.

Figura 3.3 Modelo para diseño de interfaces móviles



COMPONENTE PRIMERA: El Contexto Móvil

Es la componente más importante ya que establece las características de las restantes componentes del modelo. El contexto móvil está formado por los distintos ambientes que puede habitar el usuario de un dispositivo móvil. Las condiciones que ofrece cada ambiente son variables y determinan la forma en que un usuario interactúa con su dispositivo. Aún así, se desea que las aplicaciones sean utilizables cuando el usuario las necesita. Resulta evidente que hay condiciones que ofrecen una configuración óptima para que el usuario opere un sistema en un equipo móvil; esto tiene que ver con la comodidad que haya para visualizar la pantalla del equipo y para entrar datos, con el nivel de ruido del ambiente, con la iluminación de fondo, si se está en movimiento o no, etc.

Para lograr un uso efectivo de las aplicaciones móviles, las restantes componentes del modelo deben prever las características «no favorables» de los distintos ambientes del contexto. Si el usuario normalmente necesita determinado nivel de iluminación para operar su equipo, entonces una característica no favorable será exceso o insuficiencia de iluminación.

El contexto móvil tiene, además, una implicación social muy importante. El usuario pasa muchas horas de su tiempo «vistiendo» prácticamente un equipo portátil y éste pasa a ser parte de su vida social. Es decir, el equipo y sus aplicaciones deben ser diseñados de tal manera que no afecten ni interfieran el normal desenvolvimiento del usuario: una interfaz que constantemente emite sonidos de alertas o requiere demasiada atención sin duda provoca un impacto negativo. Para un adecuado diseño de las interfaces de usuario deben considerarse los factores sociales.

En [MS03.1] se propone una distinción para dos partes del contexto móvil: los factores sociales-psicológicos y los factores físicos-ambientales. Dentro de los factores sociales se destacan el impacto que tienen estos dispositivos desde lo individual, es decir, qué siente la persona que los utiliza, la comunicación interpersonal y la repercusión sobre el resto de la gente. Entre los factores físicos se consideran qué lleva la gente consigo y cómo se traslada, las condiciones de luz, sonido, ambientales en general; y las pérdidas de dispositivos que ocurren frecuentemente.

COMPONENTE SEGUNDA: Diseño de Formularios

La mayoría de la funcionalidad de una aplicación móvil se implementa a través de formularios. Éstos presentan al usuario el área visible de la interfaz gráfica y contienen elementos de control para la entrada y visualización de datos o aplicaciones. Cualquier ventana que se muestra es creada a partir de un formulario y pueden crease variantes de su formato.

El formulario en sí mismo no es más que un contenedor de controles. Un control permite la manipulación de datos en las aplicaciones y la interacción con la interfaz de usuario. El Compact Framework utiliza los espacios de nombres[67] *System.Windows.Forms* para el manejo de formularios y *System.Windows.Form.Control* para los distintos controles disponibles, los cuales son subconjuntos de sus espacios de nombre homónimos en el Framework completo. A diferencia de este último, hay algunas características que no han sido implementadas para el Compact Framework [MS01.1] como arrastrar elementos, soporte para la impresión, etc.

Como he comentado en el Capítulo 2, cada uno de los tres grupos de dispositivos según su plataforma (Pocket PC, Smartphone y Windows CE .NET) provee al usuario diferentes interfaces, por lo tanto se plantean distintos requerimientos para el diseño de un formulario. Recordemos las características de cada

67. El concepto de espacio de nombres (namespace) es una forma de agrupación lógica de clases de objetos que permite ordenarlas de forma jerárquica en base a criterios establecidos, a fin de que el usuario pueda localizarlas y utilizarlas con facilidad.

grupo descriptas en el Capítulo 1, en particular aquellas que tienen incidencia directa en el diseño de los formularios:

Pocket PC

- el área de visualización estándar es de 240x320 pixels, con puntos de .20 a .24 mm.
- pantalla sensible al tacto mediante un stylus[68]
- posee un panel de entrada por software llamado SIP (Software Input Panel)
- teclado emulado por software con disposición QWERTY
- botones hardware para el control de navegación, para encendido, desplazamiento, etc.

Smartphone 2003

- no posee pantalla sensible al tacto
- el display es más pequeño que el de Pocket PC, hay tres opciones distintas:
  - 176x220
  - 160x240
  - 208x240
  - tamaños de punto de 0.20 mm.
    - la entrada se realiza por medio del teclado alfanumérico del teléfono, no utiliza parte del formulario

Windows CE .NET

- aunque muchos equipos tienen una pantalla de tamaño típico de 640x450 pixels, no hay una medida estándar dada la amplia gama de dispositivos que pueden llevar Windows CE.
- Soporta una segunda pantalla y funcionamiento sin pantalla
- Soporta distintas opciones de entrada: teclado externo, mouse, SIP, pantallas sensibles al tacto y teclado integrado.

La figura 3.4 a continuación muestra una representación a escala[69] de dos formularios de Windows Mobile 2003 que corresponden a Pocket PC y a Smartphone respectivamente.

Figura 3.4 Formularios de Windows Mobile 2003

Las figuras muestran que el área de pantalla es disímil. Algunas necesitan representar un teclado en pantalla y otras no. Esto implica que los formularios para cada plataforma deben ser diseñados independientemente de acuerdo a sus características. Si se quisiera escribir una aplicación que funcione en cualquiera de las plataformas no bastaría utilizar la técnica de compilación condicional[70], pues la cantidad de elementos representados en la interfaz puede no caber en alguna de ellas (además de las diferencias en los métodos de entrada). El código que corresponde exclusivamente a la interfaz de usuario es dependiente de la plataforma y debe ser personalizado para cada una de ellas, es decir, podría escribirse una aplicación que funcione en varias plataformas previa adaptación del código que corresponde a las interfaces.

Tanto en Pocket PC como en Smartphone se habla de «vistas primarias» y se refiere a la representación visual elegida más frecuente para el diseño de las interfaces sobre formularios. En ambos casos la vista primaria es en forma de LISTAS. Los elementos que componen la interfaz se disponen verticalmente formado una especie de lista. Se puede acceder a los elementos de la lista con las teclas de desplazamiento o, si está disponible, con el stylus. En Pocket PC la disposición en pantalla tiene un aspecto similar a una

68. El stylus es un puntero (dispositivo hardware) que permite seleccionar opciones en pantalla. El usuario toma el stylus con la mano y hace clic sobre la pantalla para activar la opción que desea.
69. No están representados en tamaño real, sino que muestran la escala de relación de tamaños.
70. Consiste en incluir cláusulas que comprueben la plataforma de base en tiempo de compilación y establezcan el tamaño y otras características en base al test.

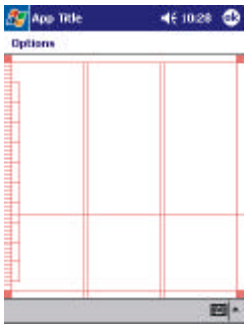
vista 2D pero en Smartphone la disposición en forma de lista se hace mucho más evidente. A continuación se exponen dos secciones de consideraciones de diseño de formularios, la primera para Pocket PC y la segunda para Smartphones. Algunos de ellos se basan en las recomendaciones de [SDK03.0] [SDK03.1].

Consideraciones para formularios en Pocket PC

- Representar en pantalla sólo las tareas principales. Puede servir la idea de la regla 80/20[2]
- Mostrar información optimizada para su lectura en el contexto móvil, más que para edición.
- Evitar incluir barras de desplazamiento en el formulario, muy comunes en Windows
- Evitar incluir menús desplegables estilo Windows
- Los controles que requieren edición o ingreso de datos, deben situarse lo más cerca posible del SIP a fin de minimizar la distancia de desplazamiento del stylus entre el control y el teclado.
- Los controles por contacto deben tener un tamaño adecuado para ser accedidos. Si se desea acceder a ellos con los dedos deben tener un tamaño no inferior a 10mm[71], si se desea utilizar el stylus deben tener al menos 5mm[2]
- Si se utilizan menús de contexto[72], deben incluirse solo las acciones más frecuentes.
- La barra de título del formulario debe tener el nombre de la aplicación.
- La cantidad de íconos que se presentan debe ser reducida. Los íconos deben ser simples, autodescriptivos (con solo verlos pueda saberse de qué aplicación se trata) y estar acompañados de un breve texto nombrando la aplicación.
- Hay que considerar que la información fluye de arriba hacia abajo, y de izquierda a derecha.

Se propone una regla basada en la métrica de 4x4: todos los controles deben mantener entre sí una «canaleta» de 4 píxel y el origen de un control siempre se corresponde con un incremento de 4 píxel. La figura 3.5 muestra la grilla resultante de esta regla.

Figura 3.5 Regla de 4x4

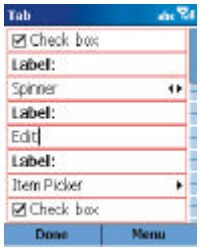


Consideraciones para formularios en Smartphone

- Al igual que en Pocket PC, se recomienda representar en pantalla sólo las tareas principales. Puede servir la idea de la regla 80/20.
- Por la disposición de lista vertical que posee, se recomienda un solo ítem por línea y una barra de desplazamiento vertical a la derecha si es necesario. Ver figura 3.6.
- La tecla *soft key*[73] izquierda suele usarse para mostrar las tareas frecuentes que usan los usuarios. También se la utiliza para implementar la función «Listo[74]», que sirve para cerrar aplicaciones o completar acciones.
- La tecla *soft key* derecha suele tener funciones asociadas de «Menú de opciones» y «Cancelar». En caso de no proveerse un menú con la tecla izquierda, esta tecla puede usarse para iniciar otras aplicaciones frecuente o bien, ser dejada sin utilidad.
- La tecla *back*[75] se usa generalmente para navegar hacia una sección anterior o para borrar caracteres escritos.

71. La regla 80/20 (basada en la idea de distribución de Pareto) indica, en este caso, que sólo deben hacerse optimizaciones para el 80% de los casos y el restante 20% puede ser eliminado.
72. Son aquellos menús que aparecen en pantalla tras hacer clic con el botón derecho del mouse en Windows y ofrecen opciones adicionales de utilidad.
73. Las teclas soft key y back fueron mostradas gráficamente en el Capítulo 1, en la sección de arquitectura de los Smartphones. Se utilizan para que el usuario interactúe con el software de aplicación y se les asocian funciones típicas de acuerdo al contexto en que se encuentren.
74. Traducido del inglés «Done», que quiere decir listo o ya hecho.
75. En inglés, «Back» significa volver atrás.

Figura 3.6 Disposición vertical en Smartphone



COMPONENTE TERCERA: Entradas

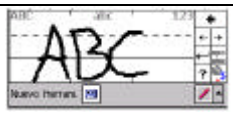
La componente de entradas está ligada a los datos que requiere una aplicación móvil y necesitan ser ingresados por el usuario. Recordemos que en la mayoría de los casos los dispositivos móviles no cuentan con un teclado estándar para el ingreso de datos. Además, el usuario se halla dentro del denominado *contexto móvil*, donde no siempre existen comodidades o condiciones apropiadas para utilizar estos equipos. Por lo tanto, una función de la componente de entrada es minimizar el ingreso de datos a través de:

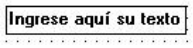
- Asegurar que la aplicación facilite al usuario el ingreso rápido y preciso de los datos
- No exigirle al usuario ingresar grandes cantidades de datos

¿Cómo facilitarle al usuario el ingreso de los datos?

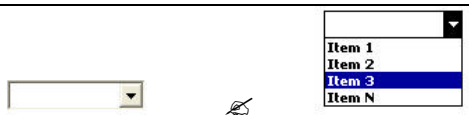
Para ello debemos conocer los controles de entrada de datos que dispone el Compact Framework y evaluar su uso, pues algunos son más adecuados que otros en aplicaciones móviles. A continuación se listan los controles de entradas más frecuentes y para cada uno de ellos se indica su nombre, la clase que lo implementa, modo de operación y observaciones para su utilización adecuada. En la parte inferior del nombre del control se halla un semáforo que representa el grado de atención que debe prestar el diseñador para no comprometer la claridad y simplicidad de la interfaz.


El diseñador debe proveer el control cuando es estrictamente necesario
El diseñador debe hacer un uso limitado y planificado del control
El diseñador puede proveer este control normalmente. No implica uso desmedido.

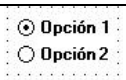
InputPanel 	 ó 
Nombre de la clase	***Microsoft.WindowsCE.Forms.InputPanel***
Modo de operación	El usuario ingresa carácter a carácter el texto que desea por medio de este teclado emulado por software, con disposición QWERTY. Se elige cada tecla haciendo clic con el stylus sobre la pantalla sensible al tacto. Este control es utilizado por otros controles para la entrada de datos manualmente. También permite la entrada por medio de la escritura natural sobre la pantalla: el usuario dibuja caracteres sobre la pantalla y al hacer una pausa, un reconocedor convierte los dibujos en los caracteres respectivos.
Observaciones	Este teclado por software es conocido como SIP (Software Input Panel), el cual se implementa en Pocket PC. Para equipos Smartphone se utiliza un teclado alfanumérico distinto. Deben implementarse entradas predefinidas siempre que sea posible, de manera de evitar que el usuario ingrese los datos manualmente con este control.

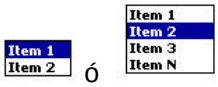
TextBox	
Nombre de la clase	***System.Windows.Forms.TextBox***
Modo de operación	El usuario ingresa carácter a carácter el texto que desea por medio del teclado previsto por cada dispositivo, software o hardware. Este control sirve tanto para la entrada manual de texto como para su visualización[76].
Observaciones	El uso de este control debe ser limitado solo a aquellos casos en que las entradas no pueden ser previstas por el sistema. Se desea que la entrada sea mínima y que vaya acompañada de algún mecanismo de validación.

CheckBox	
Nombre de la clase	***System.Windows.Forms.ChekBox***
Modo de operación	Se usa normalmente para presentar opciones de si/no, verdadero/falso, activado/desactivado. El usuario puede hacer un clic en la casilla de verificación para cambiar el estado de activación de esta opción.
Observaciones	El uso de este control es muy frecuente. Su uso moderado no invade el área de pantalla y pueden coexistir varios controles de este tipo al mismo tiempo.

ComboBox	
Nombre de la clase	***System.Windows.Forms.ComboBox***
Modo de operación	El diseñador de la interfaz de usuario conoce de antemano las posibles opciones que el usuario necesita ingresar y las carga en el control. Cuando el usuario hace clic sobre él se despliega una lista con los ítems precargados para que pueda elegir uno de ellos.
Observaciones	Este control es el primero que evidencia las técnicas de minimización de entrada y su uso es deseable. Evita que el usuario tenga que escribir datos a la vez que ahorra el escaso espacio de pantalla: la lista se despliega solamente al activar el control, el resto del tiempo permanece comprimida. Hay que tener precaución con su uso, pues una lista extensa puede traer problemas de disposición en pantalla. Para listas extensas puede usarse la técnica de Spinners expuesta posteriormente.

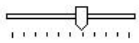
76. Las salidas se tratan en la sección aparte de componente de salida del modelo.

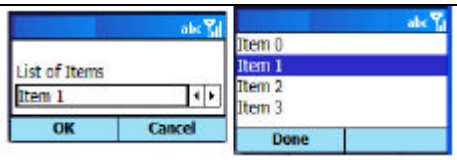
RadioButton 	
Nombre de la clase	***System.Windows.Forms.RadioButton***
Modo de operación	La interfaz muestra una serie de opciones de las cuales el usuario puede elegir solo una haciendo clic sobre la misma. Al elegir una opción, ninguna otra permanece activada.
Observaciones	El uso de este control debe ser limitado solo a aquellos casos en que las opciones son muy pocas. Esto se debe a que una única opción será la elegida y todas las otras ocupan espacio en pantalla. Si se necesitan exponer más de dos opciones es preferible utilizar un ComboBox o ListBox. Si la cantidad de opciones es extensa se recomienda el uso de Spinners.

ListBox 	
Nombre de la clase	***System.Windows.Forms.ListBox***
Modo de operación	La interfaz muestra un cuadro con opciones para elegir. Puede adaptarse de manera que se vean todas las opciones en pantalla, o bien, que se vean menos líneas y automáticamente se genera una barra de desplazamiento (en algunos dispositivos no visible) que permite iterar entre las opciones utilizando las teclas de desplazamiento del equipo o el stylus.
Observaciones	El uso de este control da la posibilidad al usuario de utilizar el espacio de pantalla según lo necesite: puede mostrarse la lista completa o solo parte de ella y desplazarse con las teclas o haciendo clic en la barra. Si la cantidad de opciones es extensa se recomienda el uso de Spinners ya que estando la lista comprimida habría que presionar demasiadas veces las teclas de desplazamiento hasta llegar a la opción deseada.

DomainUpDown	
Nombre de la clase	***System.Windows.Forms.DomainUpDown***
Modo de operación	El funcionamiento de este control se asemeja a ListBox y a ComboBox, con la diferencia que requiere muy poco espacio en pantalla (una sola línea). Para elegir una opción basta con utilizar las flechas que provee el control, presionar las teclas de desplazamiento del dispositivo o tipear el nombre.
Observaciones	Este control es muy apropiado para aplicaciones móviles por el uso muy eficiente de espacio en pantalla. Debe considerarse que el usuario no puede ver toda la lista de opciones, y si esta es extensa no resulta práctico iterarla.

NumericUpDown	 Inicial: 50 Final: 100 Incremento: 2
Nombre de la clase	***System.Windows.Forms.NumericUpDown***
Modo de operación	El funcionamiento de este control funciona igual que DomainUpDown, con la diferencia que itera entre opciones numéricas únicamente. Se debe especificar un rango de iteración formado por un número inicial y otro final, y el incremento deseado para cada salto.
Observaciones	Al igual que DomainUpDown, este control es muy apropiado para aplicaciones móviles por el uso muy eficiente de espacio en pantalla. Debe considerarse que el usuario no puede ver toda la lista de opciones, y si esta es extensa no resulta práctico iterarla.

TrackBar	
Nombre de la clase	***System.Windows.Forms.TrackBar***
Modo de operación	Este control permite la entrada de datos numéricos por medio del desplazamiento de un selector sobre una regla. Se especifica número inicial, número final y opciones de incremento para la regla inferior y para el indicador.
Observaciones	Este control es frecuente en las aplicaciones móviles que necesitan dar al usuario una rápida noción visual de una magnitud a elegir. Puede combinarse con otros controles para mostrar los valores exactos del indicador. El espacio requerido en pantalla es adaptable a conveniencia y decisión del diseñador.

Spinner 	 DomainUpDown ✍ ListBox
Nombre de la clase	No existe una clase en .NET definida específicamente
Modo de operación	Si bien esta forma de entrada de datos no está definida explícitamente en el Compact Framework, puede componerse a partir de controles conocidos como ListBox y DomainUpDown. Al combinarlos, se obtiene un control que ocupa una sola línea y puede iterarse para ver las opciones. Si el usuario presiona una tecla de acción del dispositivo (por ejemplo, Enter) se muestra un ListBox que lista varias opciones simultáneas para facilitar la entrada.
Observaciones	Esta técnica resulta muy práctica por el poco espacio requerido por el DomainUpDown. Se usa en situaciones donde la lista resulta extensa. El usuario itera sobre la lista utilizando el stylus, teclas de desplazamiento o escribiendo el componente a buscar. Si se desea ver la lista completa o detalles de un ítem, con la tecla de acción se abre un ListBox que amplía la visión de las opciones.

Todos los controles listados que ofrecen opciones predefinidas sirven como mecanismos de validación de las entradas[77] de usuarios, ya que no se permiten otras opciones más que las disponibles. Para las entradas de datos manuales es recomendable implementar mecanismos de validación por código de programación; este tema queda fuera del alcance de esta tesina. No obstante sugiero el uso de expresiones regulares[78] para verificar la validez de ciertas entradas de usuario: el Compact Framework provee las clases *Regex* y *Match* para evaluar cadenas de texto y encontrar determinados patrones. Para su uso, se requiere una referencia en el proyecto a *System.Text.RegularExpressions.dll* y la correspondiente utilización de la directiva *using*[79].

COMPONENTE CUARTA: LAS SALIDAS

La componente de salidas abarca todos aquellos controles del Compact Framework que muestran información en pantalla al usuario. La elección de los controles apropiados, la cantidad de controles presentados simultáneamente, sus características gráficas (color, tamaño, forma, movimiento) y la manera en que son accedidos determinan la usabilidad de la interfaz en el contexto móvil. En el caso de los textos, deben considerarse adicionalmente los tipos de fuentes, la extensión del texto a presentar y el empleo de una semántica simple que permita una fácil lectura y una rápida operación de la aplicación.

Consideraciones para el uso de colores

Al momento de diseñar cualquier componente visual de salida debe pensarse en el uso de colores como elemento indicador, es decir, no usar colores arbitrariamente sino tratar de ofrecer un valor extra al usuario por medio de los mismos. Algunos colores pueden ser utilizados para resaltar partes importantes y captar la atención del usuario. El uso de la técnica de altos contrastes ayuda a la legibilidad en el contexto móvil.

77. Los mecanismos de validación de entradas son construcciones de software que permiten verificar la validez de los datos ingresados por el usuario, a fin de evitar numerosos errores que son consecuencia de estas entradas espurias.
78. Las expresiones regulares son generadores de lenguajes regulares que pueden ser utilizados como reconocedores de los mismos. Permiten validar entradas de usuario por medio del reconocimiento de determinados patrones e informar cuando una cadena de texto no cumple la especificación del patrón.
79. La directiva using se utiliza en el Compact Framework para indicar al compilador el uso y referencia de algunas clases.

Algunos dispositivos de pantalla difieren según sus características de hardware, por ejemplo, en la profundidad de color, tamaño de punto, etc. Esto debe ser tenido en cuenta porque el diseñador puede haber empleado colores en su herramienta de desarrollo que luego no tendrán la misma correspondencia de color en el dispositivo real. Para ello hay que conocer las características del dispositivo objetivo.

Es muy recomendable no utilizar colores generados a partir de mezclas. En vez de ello, lo mejor es utilizar los nombres de colores estándar que el Compact Framework provee a través de la clase *Color*; esto ayuda a mantener una consistencia con el diseño de las aplicaciones Windows y facilita la eventual portabilidad entre dispositivos.

Consideraciones para el uso de texto

Se parte de la base que las personas que utilizan dispositivos móviles no cuentan con comodidades ni tiempo suficiente para leer grandes cantidades de textos. Además, el espacio en pantalla es crítico. Tanto la **cantidad** de texto, como la **semántica**, deben tener especial atención por parte del diseñador. No es tarea trivial la selección del texto a incluir en la interfaz, pues las restricciones son varias. A continuación se listan una serie de técnicas que pueden facilitar la tarea de diseño cuando hay textos involucrados:

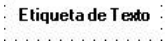
- El uso de expresiones informales suele acortar los textos. Por ejemplo, «No es posible conectar» en vez de «Su equipo no puede acceder al servidor en este momento»
- Suprimir texto superfluo. Por ejemplo, «Batería baja», en vez de «Su equipo está operando en niveles bajos de batería»
- Utilizar símbolos para dar indicaciones. Por ejemplo: «Elija Edición>Copiar» en vez de «Haga clic en el menú Edición y seleccione la opción Copiar»
- Utilizar negritas solo para encabezados, con el fin de facilitar la lectura y no para enfatizar significados.
- No se recomienda el uso de itálicas.
- No utilizar los dos puntos «:» para encabezados o títulos de columnas.
- Utilizar términos claros y sencillos de comprender.
- Evitar el uso de: tecnicismos, formas coloquiales u ofensivas.
- El uso de abreviaturas puede resultar ambiguo para el usuario.
- El formato del texto debe ser consistente con el estilo que se quiere dar al formulario y debe estar en armonía con el resto de los elementos presentados en la interfaz.

Las fuentes de uso común para Pocket PC y Smartphone son Tahoma de 8 puntos y Nina de 11 puntos respectivamente. (Este es un texto escrito en Tahoma de 8 puntos)

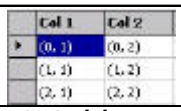
[SDK03.0] y [SDK03.1] recomiendan además lo siguiente:

- Para el uso de acrónimos deben tenerse en cuenta cuestiones regionales, legales, de consistencia y no explicar su significado en la ayuda del dispositivo. Usar ese espacio para cosas más útiles y delegar esta tarea al manual de usuario impreso.
- El uso de texto instructivo debe ser limitado a pocas palabras y su fin es evitar que el usuario tenga la necesidad de acudir a la ayuda.
- La interfaz debe pasar una evaluación de características regionales si se desea que opere en más de un país. Dicha evaluación consiste en examinar el uso de unidades, magnitudes, traducciones y términos adecuados para cada región.

A continuación se listan los controles de salida más frecuentes y para cada uno de ellos se indica su nombre, la clase que lo implementa, modo de operación y observaciones para su utilización adecuada. Tal y como se hizo anteriormente para las entradas, en la parte inferior del nombre del control se halla un semáforo que representa el grado de atención que debe prestar el diseñador para no comprometer la claridad y simplicidad de la interfaz.

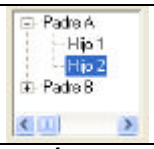
Label 	
Nombre de la clase	***System.Windows.Forms.Label***
Modo de operación	Se le asigna una cadena de caracteres y se muestra de manera estática, es decir, su contenido no puede ser modificado durante la ejecución.
Observaciones	Puede ser utilizado tanto para textos muy pequeños con el objetivo de poner rótulos o etiquetas lingüísticas o para textos más extensos tales como instructivos.

TextBox 	
Nombre de la clase	***System.Windows.Forms.TextBox***
Modo de operación	Se le asigna una cadena de caracteres y se muestra su contenido durante la ejecución. Su contenido puede ser estático (en modo de sólo lectura[80]) o dinámico.
Observaciones	Debe ser utilizado únicamente para textos/números muy pequeños que no excedan una línea simple.

DataGrid 	
Nombre de la clase	***System.Windows.Forms.DataGrid***
Modo de operación	Este control muestra datos provenientes de una tabla de datos o de una colección[81]. Genera una representación en forma de grilla que puede personalizarse respecto a sus columnas, títulos, ancho y eventos que se generen en la misma. Los usuarios pueden elegir ítems utilizando las teclas de desplazamiento del dispositivo o tipeando su nombre.
Observaciones	Este control ocupa mucho espacio de pantalla y debe ser usado en casos de necesidad específica. Los anchos de columna incorrectamente ajustados pueden crear problemas para la visualización de los datos. No permite la edición dinámica, aunque puede implementarse manualmente. No está soportado en Smartphone.

ListView 	
Nombre de la clase	***System.Windows.Forms.ListView***
Modo de operación	Este control permite mostrar listas de datos dentro de un cuadro que posee una barra de desplazamiento. La lista consta de ítems y cada ítem posee una etiqueta lingüística y opcionalmente una imagen asociada. Posee una propiedad para elegir las vistas posibles: íconos grandes, íconos pequeños, lista y detallada. Es posible iterar sobre la lista utilizando las teclas del equipo.
Observaciones	Para los dispositivos móviles se recomienda utilizar la vista detallada, con íconos de 16x16 píxel y el encabezado de la columna oculto. Para Smartphone, deben usarse a los sumo 3 columnas y debe usarse pantalla completa.

80. El Compact Framework permite cambiar las propiedades de algunos controles para que no sea posible la entrada de datos en tiempo de ejecución. Dicha propiedad se denomina *ReadOnly*.
81. Compact Framework ofrece un conjunto de clases en el espacio de nombres *System.Collections* que permite a los desarrolladores trabajar con colecciones o conjuntos de datos y manejarlos de manera simple de acuerdo a la naturaleza de su estructura de datos.

TreeView	
Nombre de la clase	***System.Windows.Forms.TreeView***
Modo de operación	Este control exhibe de manera jerárquica una colección de datos. Utiliza la clase *TreeNode* del Compact Framework para armar una representación en forma de árbol. Ofrece múltiples opciones para formatear la salida: puede incluir imágenes, líneas para los nodos, se expande y contrae dinámicamente.
Observaciones	Se recomienda no realizar representaciones expandidas que superen los 12 nodos (incluyendo padres e hijos) para Pocket PC y 8 nodos para Smartphone. Este último no muestra las líneas punteadas que unen los nodos ni los íconos de expandir/contraer: por medio de la propiedad *AutoScroll* el control se expande automáticamente. Para Smartphone debe usarse en pantalla completa.

PictureBox	
Nombre de la clase	***System.Windows.Forms.PictureBox***
Modo de operación	Este control es contenedor y exhibidor de una imagen. Se le asocia un archivo gráfico y este es mostrado en el área establecida para el control.
Observaciones	Debe prestarse mucha atención al tamaño que ocupa el archivo de la imagen ya que ocupará espacio en memoria, y a la dimensión de la imagen que utilizará espacio en pantalla. No se recomienda el uso de archivos animados porque consumen recursos de procesamiento y restan tiempo de autonomía a la batería del equipo.

COMPONENTE QUINTA: La comunicación del usuario con el sistema

Esta última componente del modelo tiene como objetivo dar algunas pautas acerca de cómo las aplicaciones móviles deben brindar información a los usuarios y cómo los usuarios operan las aplicaciones a través de controles. No se trata de la entrada/salida de datos, los cuales fueron tratados en la componente anterior, sino de notificaciones que la aplicación o el sistema necesitan dar a conocer a sus usuarios y de los controles que las aplicaciones proveen para su uso.

Consideraciones para el uso de mensajes de error

Los mensajes de error son de carácter informativo para el usuario y por ellos deben ser escritos de forma concisa. Se desea que cada mensaje de error tenga al menos 3 partes:

1. identificación del problema
2. indicación de la causa que lo generó (si sirve de ayuda)
3. sugerencia sobre cómo resolverlo

Los errores que impiden al usuario realizar sus tareas principales deben ser acompañados de la sugerencia de resolución indicada en el punto tercero, ya que el usuario móvil no siempre lleva consigo el manual de operación y puede serle difícil solicitar soporte desde su ubicación. En el mismo mensaje de error puede

proveerse un enlace hacia la sección a la que debe dirigirse el usuario para resolver el inconveniente. Debe utilizarse la menor cantidad de texto posible (puede seguir las sugerencias propuestas para textos descriptas anteriormente en la componente de salida).

Los mensajes de error pueden implementarse a través de la clase *MessageBox* o de cuadros de diálogo utilizando el método *ShowDialog( )* de la clase *Form* cuando se requiere confirmación.

Algunas recomendaciones adicionales son:

- poner en mayúsculas los títulos de los mensajes de error
- si los mensajes incluyen comandos, ponerlos en letras negritas
- para dar mensajes de alerta, se recomienda poner la palabra «Atención» solo en aquellos casos donde la decisión que tome el usuario determine el resultado final.
- no se recomienda el uso de signos de admiración
- no personificar a las aplicaciones para dar la idea de que *sienten* o *piensan*. Ejemplo: «Me estoy quedando sin batería» no es un mensaje adecuado. En vez de esto, debe usarse «Batería Baja. Se recomienda reemplazarla ahora»
- evitar el uso de expresiones tales como *resetear*, *se colgó*, *se tildó*, etc. Deben emplease términos que describan lo más exacto posible la situación y las acciones a realizar.
- solicitar confirmación únicamente para aquellas acciones que no sea posible la vuelta atrás (por medio de alguna función similar a *deshacer*)

Documentación en línea de las aplicaciones

La documentación en línea consiste en información de ayuda o soporte incorporada en la aplicación, o bien, es provista en archivos aparte y se sitúa en una carpeta visible para que el usuario la identifique. Para la gran mayoría de las aplicaciones, contar con documentación en línea es de suma utilidad para los usuarios. Como sabemos, durante el desarrollo de aplicaciones móviles tratamos de ahorrar el máximo posible espacio en memoria a causa de las limitaciones ya conocidas. En los dispositivos que poseen una capacidad relativamente mayor, será posible generar algunos documentos de soporte para aquellas aplicaciones que así lo requieran. En el caso de los más restringidos, como ocurre con los Smartphones, no es recomendable incluir documentación en línea pero pueden utilizarse pequeños instructivos en pantalla para las situaciones que así lo requieran.

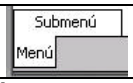
En Pocket PC, se estila (aunque no es una norma) escribir documentos de ayuda utilizando el lenguaje HTML[82] estándar, en los que se incluyen gráficos, archivos con extensión *.htm* y palabras claves para las búsquedas. Cabe aclarar que los archivos *.chm* (HTMLHelp) no funcionan en Windows CE. Las ayudas HTML por convención tienen los siguientes controles:

- View (ver). Muestra el primer nivel de la ayuda HTML construida de manera jerárquica y una lista de todos los componentes de ayuda creados.
- Find (Buscar). Abre un cuadro de diálogo que permiten ingresar palabras claves para la búsqueda dentro del dominio de la ayuda.
- Adelante. Permite navegar hacia delante.
- Atrás. Permite volver hacia la página visitada inmediatamente anterior.

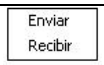
A continuación se listan los controles de operación más frecuentes y para cada uno de ellos se indica su nombre, la clase que lo implementa, modo de operación y observaciones para su utilización adecuada. A fin de seguir la metodología anteriormente empleada, en la parte inferior del nombre del control se halla un semáforo que representa el grado de atención que debe prestar el diseñador al control en cuestión para no comprometer la claridad y simplicidad de la interfaz.

82. En inglés Hypertext Markup Language, lenguaje por marcación de hipertexto, ampliamente utilizado y de uso estándar para la generación de páginas web.

Button	
Nombre de la clase	***System.Windows.Forms.Button***
Modo de operación	Este control se crea en pantalla y puede ser activado por medio de las teclas de desplazamiento o a través de la pantalla sensible al tacto. Tras su activación se genera un evento[83] cuyo manejador incluye el código que la aplicación debe realizar.
Observaciones	¿Porqué este control tiene semáforo rojo? La respuesta es que los dispositivos de Pocket PC y Smartphone suelen tener botones hardware que complementan la funcionalidad de la aplicación. Por ende, siempre que sea posible debe hacerse uso de tales botones hardware y dejar este control para casos especiales. Se recomienda un tamaño estándar de $5mm^2$ para botones que vayan a ser activado con el uso de stylus y de $10mm^2$ para activaciones dactilares. No recomiendo esta última dado que los aceites naturales de la piel van ensuciando la pantalla y la limpieza constante del equipo no es práctica en el contexto móvil.

Menu	
Nombre de la clase	***System.Windows.Forms.MainMenu y MenuItem***
Modo de operación	Los menús desplegables permiten acceder a grupos de funciones relacionadas que se organizan por categorías. Cada función se implementa mediante un *MenuItem* contenido en un *MainMenu*. Al hacer clic en este último se despliegan los ítems que contiene. Permite el anidamiento de menús.
Observaciones	Para dispositivos con pantalla pequeña, como Pocket PC y Smartphone, se recomienda el uso de menús únicamente si se disponen de tal forma que no produzcan desorden en pantalla, esto es, que la cantidad de ítems sea menor a 15 y a 9 respectivamente. En el uso de submenús se recomienda que no supere el primer nivel de anidamiento, o sea, que existan menús y submenús de primer nivel. Hay que tener presente que es difícil navegar una aplicación cuyas funciones se encuentran sumergidas en una jerarquía de menús. Los menús se sitúan en la parte inferior en Pocket PC.

83. El Compact Framework maneja un modelo similar al esquema productor-consumidor para el manejo de controles de interfaz. Un evento es una acción o condición que se genera y determina que éste debe ser administrado por un manejador, el cual incluye el código necesario.

ContextMenu 	
Nombre de la clase	***System.Windows.Forms.ContextMenu***
Modo de operación	Los menús de contexto permiten desplegar opciones adicionales de acuerdo al entorno de operación actual. Son similares a aquellos que aparecen en Windows de escritorio haciendo clic con el botón derecho del mouse. En las otras plataformas puede asociarse su visualización a alguna tecla de acción del dispositivo.
Observaciones	Se emplea este control para implementar funciones de mucha utilidad pero de uso relativamente frecuente. De ser usado, incluir pocas opciones, especialmente en dispositivos de pantalla pequeña.

ProgressBar 	
Nombre de la clase	***System.Windows.Forms.ProgressBar***
Modo de operación	Este control provee una rápida noción visual del estado de evolución de una tarea. Se programa para que, a medida que la tarea de va completando, el control muestre de forma gráfica este cambio.
Observaciones	Es un control muy apropiado para aplicaciones móviles dada la facilidad de visualización que ofrece. Se debe utilizar algún control extra para que muestre información adicional acerca de las magnitudes de completitud.

3.3 Consideraciones de Seguridad

Contar con una buena infraestructura en tecnología de la información (IT) se ha convertido en una fuente de ventaja competitiva para las organizaciones. Debidamente administrada, IT puede lograr diversas mejoras que ayudan a contribuir a los objetivos organizacionales. Aplicada a áreas específicas puede conducir a reducción de costos de producción, de operación, eficiencia en procesos, mejora en las relaciones con los clientes, incrementos en las utilidades, etc.

La incorporación de dispositivos móviles y tecnologías inalámbricas en la empresa permite que los sistemas locales sean accesibles remotamente de manera muy conveniente. Así, los sistemas que se operaban exclusivamente en un área confinada a los edificios de la organización ahora pueden ser extendidos con el uso de aplicaciones móviles. Los pequeños dispositivos resultan mucho más prácticos que los notebooks[84] para la recolección de datos y para el traslado en el contexto móvil.

No obstante, todo elemento que forma parte de IT requiere un análisis previo de los riesgos potenciales que introduce el uso de tales tecnologías. Sabemos por definición que la palabra riesgo significa la probabilidad de que un suceso adverso ocurra y que tenga un impacto que genere algún perjuicio. El riesgo entraña al menos tres componentes principales, la causa que lo provoca, su probabilidad de ocurrencia y el costo de prevenirlo o mitigarlo.

Con el uso de dispositivos móviles existen riesgos. Entre ellos encontramos riesgos de integración con tecnologías anteriores, sean sistemas u otros dispositivos, de no alcanzar la funcionalidad requerida, de incrementar los costos en detrimento de las ganancias esperadas, de producir un impacto negativo sobre el personal que los utiliza, etc.

84. En relación a algunas tareas, no se plantea al dispositivo móvil como un elemento sustito de las computadoras portátiles.

Un riesgo muy importante a considerar es el riesgo en la **seguridad**. Cuando hablamos de seguridad en ámbito de los dispositivos móviles nos referimos a la seguridad en la información, probablemente uno de los recursos más valiosos de toda organización (también aplicable a la información de la persona individual).

La seguridad en la información puede definirse a través de cinco conceptos básicos:

1. confidencialidad
2. integridad
3. autenticación
4. no repudio
5. disponibilidad

La **confidencialidad** expresa que la información sea accesible sólo por las personas o sistemas pretendidos. Cualquier acceso no autorizado viola los principios de confidencialidad. La **integridad** se refiere a que la información no sufra pérdidas ni alteraciones indeseadas. La **autenticación** tiene que ver con validar efectivamente si la persona o sistema es quien dice ser, con el fin de negar los accesos no autorizados. El concepto de **no repudio** está asociado a la comprobación de la autoría de una pieza de información determinada, o sea, el manejo de firmas personales. Por último, la **disponibilidad**, expresa que la información esté disponible cuando el usuario la requiere y su acceso no se vea interrumpido intencionalmente con fines opuestos a los empresariales.

A continuación, la tabla 3.6 indica algunos ejemplos de sucesos indeseables asociados a los conceptos de seguridad en la información.

Tabla 3.6. Ejemplos de sucesos indeseables para cada concepto de seguridad

Concepto	Ejemplos de sucesos adversos
confidencialidad	se intercepta una comunicación WiFi[85] de manera ilegítima
integridad	se cambia el contenido de un mail interceptado de manera ilegítima
autenticación	passwords adivinados o mal obtenidos
no repudio	una persona niega un mensaje que realmente envió
disponibilidad	el enlace de red fue saboteado y un PDA móvil no puede conectarse

Los dispositivos móviles introducen algunos riesgos debido a su naturaleza portátil: [MS01.1] y [MS01.3]

- pueden contener información acerca de contactos, clientes o personas y no es deseable que caiga en manos de la competencia
- las aplicaciones móviles requieren datos. Si esos datos se pierden, dañan o son accedidos por las personas incorrectas pueden generar grandes perjuicios
- en el contexto móvil pueden ocurrir numerosos incidentes: un dispositivo puede caerse, mojarse, perderse, ser robado, olvidado, dejar de funcionar por desperfectos técnicos, etc.
- en algunos países pueden surgir inconvenientes legales si la información de un dispositivo es robada, sea el caso de información médica, personal, etc.

Por los riesgos mencionados, las empresas que desean incorporar el uso de dispositivos móviles deben definir una política de seguridad de manera que cada integrante tenga en claro su responsabilidad. Esto adquiere importancia de acuerdo al tipo de información que se desea almacenar en el equipo, o de las posibilidades de acceso a sistemas organizacionales que el dispositivo ofrezca.

Si bien no podemos controlar la ocurrencia de sucesos adversos, los desarrolladores de aplicaciones móviles deben tomar precauciones para minimizar el impacto si tales sucesos ocurren. La empresa Microsoft realiza una serie de sugerencias para las políticas de las organizaciones que se ocupan de la seguridad de sus datos [MS01.2], las cuales he complementado a continuación:

Empleo de passwords o claves de seguridad

Los dispositivos deben tener activada la opción de solicitud de password al encenderlo. Si el equipo no es operado por un lapso determinado debe apagarse automáticamente y volver a solicitar la clave de seguridad. Pueden emplearse otros mecanismos de autenticación como tarjetas de seguridad o técnicas biométricas[86]. No deben almacenarse los códigos de seguridad en el dispositivo, especialmente aquellos que

85. WiFi hace referencia a un conjunto de estándares de la familia 802.11x para la comunicación inalámbrica entre computadoras y dispositivos.

86. Las técnicas biométricas permiten reconocer rasgos humanos que son propios de cada persona, por ejemplo las huellas digitales, el iris, etc.

permiten el acceso a las redes de la organización. La administración de claves de seguridad no es una tarea trivial: debe planificarse la asignación, emplear las mejores prácticas[87] para la conformación de las claves, realizar cambios periódicos, etc.

Pocket PC, a partir de las versión 2002 incorpora una aplicación que permite definir claves de acceso simples de 4 dígitos o claves complejas alfanuméricas que deben combinar minúsculas, mayúsculas y signos de puntuación.

Utilizar software antivirus

Existen virus desarrollados que operan en las plataformas móviles y pueden contagiarse a los sistemas remotos por medio de las redes de comunicaciones. Por esto deben emplease antivirus en los dispositivos.

Criptografía

La criptografía tiene por objetivo hacer indescifrable la información que es capturada de manera ilegítima. Puede emplease en el dispositivo localmente para proteger datos confidenciales o para cifrar comunicaciones. Puede realizarse criptografía por software para las comunicaciones y complementarse con criptografía por hardware para información que se almacena localmente (para ello existen tarjetas que se conectan directamente al dispositivo).

Para encriptar datos, un dispositivo debe soportar los estándares de la industria de 40/128 bits, o emplear estándares de la organización que puedan funcionar conjuntamente con los estándares de la industria (como CryptoAPI).

Uso de memoria flash ROM[88]

La tecnología flash ROM permite que muchos fabricantes incorporen el sistema operativo y las aplicaciones principales a este tipo de memoria más segura, ya que no puede ser escrita por aplicaciones de usuario. Ofrece más protección antivirus en la medida en que éstos no pueden escribir determinadas zonas de la memoria ROM. Por otro lado, si ocurren problemas con la alimentación del equipo los datos permanecen almacenados. Existen tarjetas externas que se conectan al equipo y mantienen los datos independientemente. Es muy importante que los datos contenidos en las tarjetas externas se encuentren encriptados, ya que las tarjetas pueden ser olvidadas o sustraídas.

Las políticas y tecnologías recientemente citadas deben aplicarse a las soluciones móviles en tres puntos claves: [MS01.3]

1. acceso al dispositivo
2. acceso a la información almacenada en el dispositivo/periféricos
3. acceso a redes de datos

Criptografía en Compact Framework

En la sección de «Presentación del Compact Framework» del capítulo 2 he expuesto que no se ha incluido soporte para la criptografía en las librerías FCL. Si bien no surge precisión acerca del motivo, es de suponer que se debe a que las clases ya implementadas en el Framework estándar pueden haber resultado ineficientes para la ejecución en un dispositivo móvil. No obstante, una alternativa ante esta limitación es emplear la CryptoAPI que ofrecen las plataformas basadas en Windows CE. Esta API provee una forma consistente de encriptar datos para su envío internacional. No pertenece al Compact Framework y por lo tanto trabaja con código no administrado. De todas formas, es posible que las aplicaciones con código administrado hagan uso de la API a través de *P/Invoke*[89].

4.1 Concepto de Modelo de Computación Distribuida

Formas de procesamiento: camino hacia la computación distribuida

Una empresa puede organizar su función de procesamiento de datos de manera centralizada, donde el procesamiento de datos ocurre en una computadora o un grupo de computadoras centrales. En **sistemas centralizados** completamente existen:

- Computadoras centralizadas en lugares físicos, generalmente equipos poderosos.
- Procesamiento centralizado, donde todas las aplicaciones se ejecutan en un equipo central

87. Hay organizaciones que definen un conjunto de políticas para la adecuada conformación de passwords. Por ejemplo: no emplear palabras del diccionario, números de DNI, fechas de cumpleaños, etc.

88. En inglés, Read Only Memory significa memoria de sólo lectura. El usuario o las aplicaciones no pueden escribir directamente sobre la memoria; sólo se permiten lecturas.

89. *P/Invoke* (Platform Invoke) es una capa de interoperabilidad que posee el Compact Framework que permite al código administrado hacer invocaciones a librerías de enlace dinámico (DLLs). Esta funcionalidad se provee en el espacio de nombres *System.Runtime.InteropServices*

· Datos centralizados, todos los datos se almacenan en una computadora central

Una estrategia distinta consiste en contar con esquema de **procesamiento de datos distribuido**[90], donde computadoras menos poderosas que las empleadas en sistemas centralizados se dispersan en distintos lugares físicos, dentro y fuera de la organización. La distribución de las computadoras tiene el fin de realizar el procesamiento de datos teniendo en cuenta factores geográficos, operacionales y económicos. Los equipos se encuentran conectados mediante algún tipo de red de datos. Este esquema de procesamiento tiene una serie de ventajas:

· *Disponibilidad*. Se intenta minimizar el impacto en la red que ocasiona el no funcionamiento de un equipo o sistema.
· *Crecimiento gradual*. Es posible reemplazar aplicaciones o sistemas sin provocar un impacto negativo en el sistema completo. En sistemas centralizados, si se incrementa la carga de trabajo del equipo principal progresivamente puede implicar actualizar el hardware o piezas de software (implica muy altos costos).
· *Nivel de respuesta y productividad*. Los recursos locales pueden ser administrados de forma que satisfagan directamente las necesidades de performance de sus usuarios locales, en oposición a los recursos que deben satisfacer a un gran grupo de usuarios de toda una organización. Suelen lograrse mayores niveles de productividad debido a los menores tiempos de respuesta.

Con el fin de implementar funcionalidad de cliente/servidor[91] las organizaciones emplearon la técnica de **paso de mensajes distribuidos**[92]. Así, un equipo cliente que requiere algún servicio le envía un mensaje a un equipo servidor con la solicitud de servicio. El servidor recibe la solicitud, realiza las acciones solicitadas y envía un mensaje de vuelta con la respuesta. De forma general, la comunicación consiste en mensajes de *solicitud* y *respuesta*. Una solicitud es utilizada por un proceso que requiere enviar un mensaje e incluye parámetros tales como destinatario y contenido del mensaje. Los mensajes son enviados por una conexión de red y son conducidos por ésta hasta el módulo de paso de mensajes que examina los parámetros.

Las llamadas a procedimientos remotos, **RPC**[93], representan una variación del modelo de paso de mensajes. Este esquema de encapsulamiento de comunicaciones en sistemas distribuidos ha sido ampliamente aceptado y utilizado. La idea central de RPC es permitir a programas situados en diferentes máquinas interactuar utilizando una semántica simple de *invocaciones* y *respuestas*, y así dar la idea de que ambas aplicaciones se encuentran en el mismo equipo. Es la base para acceder a **servicios** remotos. Ha sido ampliamente utilizado debido a que:

· La invocación de procedimientos (no remotos, sino en su sentido más general) ha sido considerado como un mecanismo de abstracción fácil de comprender y, por ende, conocido por los desarrolladores.
· Se pueden especificar y documentar las interfaces, las cuales se componen de procedimientos formados por una lista de parámetros con tipos. Los programas distribuidos pueden realizar chequeos estáticos de las interfaces.
· La ventaja de especificar interfaces estándar es que los desarrolladores pueden crear módulos de cliente/servidor y ser utilizados en distintos equipos y sistemas operativos con pequeñas adaptaciones (idealmente). Además, existen herramientas que generan código automáticamente a partir de las especificaciones de interfaces.

Como se dijo anteriormente, el mecanismo de RPC se basa en un principio fácil de comprender, sin embargo la experiencia ha demostrado que el desarrollo de aplicaciones que emplean RPC resultó dificultoso en algunos casos. Por ello, se crearon otros recursos tecnológicos para intentar ocultar la complejidad de RPC utilizando objetos, lo que originó distintas variantes de *ORPC* (Object Oriented RPC).

Muchas organizaciones con necesidades de **interoperabilidad** en sus sistemas visualizaron que la **Web** era un medio muy apropiado para llevarla a cabo.

La necesidad de sistemas interoperables

Hay organizaciones que poseen información que se encuentra restringida en sistemas internos aislados, los cuales no fueron diseñados para intercambiar información con otros sistemas. Si tales piezas de infor-

90. Aún no estamos hablando del modelo de computación distribuida.
91. La característica principal de una arquitectura cliente/servidor es que un cliente solicita servicios a un servidor a nivel de aplicación, sin importar los detalles de conexiones de red subyacentes. Así, las plataformas y sistemas operativos de clientes y servidores pueden ser diferentes.
92. En inglés, *Distributed Message Passing* [STA01]
93. RPC es un acrónimo en inglés de *Remote Procedure Call*, invocación a procedimientos remotos. Es una técnica en la que dos programas, que se ejecutan en máquinas diferentes, interactúan por medio de llamadas/respuestas a procedimientos remotos. Ambos programas se comportan como si estuvieran ejecutándose en la misma máquina.

mación pudieran ser compartidas o intercambiadas con otros sistemas, el valor de la información podría aumentar notablemente. Conectar sistemas aislados requiere inversiones de dinero y lleva tiempo. En parte se debe a que cada organización emplea tecnologías distintas para implementar sus soluciones, o incluso en el ámbito de una misma empresa conviven distintas tecnologías de hardware, de sistemas operativos, de protocolos de comunicaciones, etc. Por otra parte se debe a que los sistemas no fueron «pensados» y diseñados para la interacción desde el principio.

La interoperabilidad viene incrementando su importancia en gran medida desde la última década. Se identifican dos áreas principales donde la interoperabilidad es deseable: [MS01.5]

1. Integración de aplicaciones empresariales (EAI[94])
2. Interacciones negocio a negocio (B2B*i* [95])

La primer área, EAI, es una forma de interoperabilidad interna. Se desea comunicar entre sí los distintos sistemas que forman parte de la organización. La segunda, representa las interacciones de negocio entre empresas diferentes.

Considerando el hecho que los servidores y navegadores Web se encuentran disponibles para la mayoría de las plataformas y que forman parte de la infraestructura tecnológica de muchas organizaciones, la interoperabilidad se ve facilitada aprovechando la Web. Las **aplicaciones Web** se construyen en torno a navegadores y constituyen una nueva forma de computación distribuida. El usuario accede a la aplicación por medio de la Web, con un navegador, como si se tratase de un sitio Web común. La limitación principal es que se requiere el humano para su ejecución, esto es, las aplicaciones Web tienen una interfaz gráfica orientada al usuario y toda la funcionalidad de la aplicación se accede desde ese punto único (un componente de software no puede utilizar la aplicación). Por esta causa, las aplicaciones distribuidas no pueden funcionar como componentes de aplicación distribuidos.

Para poder aprovechar las ventajas de múltiples sistemas comunicados entre sí debe existir un modelo que defina la infraestructura necesaria. El **modelo de computación distribuida** es una forma estándar de llevar a cabo la interacción de sistemas y está formado por cuatro elementos fundamentales:

- aplicaciones que requieren exponer o solicitar funcionalidad
- servicios Web[96]
- protocolos y estándares abiertos[97] de comunicaciones
- Internet

El principio es: los elementos del modelo se relacionan para lograr que las aplicaciones trabajen de manera conjunta sin importar dónde se sitúan ni cómo fueron implementadas.

Los servicios Web representan una evolución tecnológica de componentes distribuidos como RPC y son las unidades fundamentales para alcanzar los objetivos del modelo de computación distribuida. Los protocolos y estándares permiten que los servicios Web sean la plataforma de integración de aplicaciones heterogéneas a través de Internet.

4.2 XML Web Services

Introducción a XML

XML se utiliza como un medio independiente de plataforma de hardware y software para la representación y descripción de datos. Permite «ocultar» las diferencias entre plataformas ya que es un estándar que se basa en *texto* para la representación de los datos.

Los documentos XML utilizan una combinación de *elementos* y *atributos* para representar la información de manera jerárquica.

XML es un acrónimo en inglés para *Extensible Markup Language*, que significa lenguaje por marcación extensible. [W3C01]

¿Qué significa que sea por marcación?

Un lenguaje por marcación especifica un conjunto válido de notaciones para definir la estructura y utilización de los datos. Las marcas se realizan mediante *tags*, que son cadenas de caracteres encerradas entre los símbolos < >.

94. Acrónimo en inglés de *Enterprise Application Integration*
95. Acrónimo en inglés de *Business to Businees Interactions*
96. En inglés, *Web Services*, concepto que se describe en la siguiente sección.
97. Los protocolos abiertos publican abiertamente sus procedimientos de transmisión de datos e interfaces para que distintas empresas desarrolladoras o fabricantes de hardware puedan hacer sus propias implementaciones, con el objetivo de lograr la interoperabilidad de sistemas heterogéneos.

¿Qué significa que sea extensible?

Los tags que emplea XML no están definidos a nivel de lenguaje, es decir, el desarrollador debe definir sus propios tags y la estructura del documento. HTML en cambio si tiene un conjunto predefinido de tags que el autor debe emplear. Además, XML puede ser considerado un metalenguaje ya que es posible definir nuevos lenguajes de marcación a través de él.

¿Cuál es la diferencia entre XML y HTML?

La diferencia principal es que XML fue pensado para llevar[98] datos. No son lenguajes sustitutos sino complementarios. Si bien ambos son lenguajes por marcación, XML fue diseñado para describir datos y se enfoca sobre qué son los datos. En cambio, HTML fue diseñado para mostrar información y se enfoca en cómo se presentan los datos.

Servicios Web XML

Los servicios Web XML son unidades discretas de código que manejan un conjunto de tareas. Las aplicaciones que los utilizan exponen una serie de servicios a sus usuarios o sistemas, a través de protocolos Web abiertos. Representan una nueva «plataforma» en la que desarrolladores pueden construir aplicaciones distribuidas teniendo a la interoperabilidad como característica de máxima prioridad [MS01.5].

Los servicios Web XML extienden el principio del modelo de computación distribuida: no sólo las aplicaciones pueden trabajar de manera conjunta sino que, las personas también pueden comunicarse con sistemas y con otras personas. Tienen las siguientes características: [MS01.4]

- Permiten que programas escritos en distintos lenguajes sobre plataformas diferentes se comuniquen entre sí mediante el uso de estándares
- Se ejecutan sobre protocolos y estándares ampliamente utilizados como TCP/IP, HTTP y XML. La ventaja de esto radica en que las empresas que ya cuentan con una infraestructura Web han manejado estos estándares, cuentan con experiencia y el costo de introducir el uso de servicios Web XML es significativamente menor comparado a tecnologías anteriores.
- Proveen mecanismos para describir sus interfaces de manera que los clientes puedan usar esta información para comunicarse con ellos.
- Pueden ser registrados y localizados universalmente. Los usuarios que requieran sus servicios tienen así una forma de hallarlos fácilmente.

Tabla 4.1. Ventajas y consideraciones principales del uso de XML Web Services

ventajas	consideraciones
Utilizan protocolos estándar de la industria y no se requiere una infraestructura *adicional porque funcionan sobre Internet.*	Para tener acceso a los servicios Web se requiere de una conexión de red.
Ofrecen servicios entre sistemas no homogéneos ocultando los detalles internos de implementación	El formato que define XML no es compacto como otros formatos, por ejemplo: codificaciones binarias.
Las tareas que requieren uso de recursos no disponibles pueden ser realizadas en otros equipos/sistemas y solicitar los resultados a través de estos servicios.	Para transferencias de grandes cantidades de datos se requieren redes de datos con mayores anchos de banda.
Se han definido protocolos para poder localizar y utilizar el uso de servicios Web ya desarrollados por terceras partes.	La prestación de un servicio Web puede requerir tiempos prologados en algunos casos. Las aplicaciones deben diseñarse de manera que realicen otras acciones mientras de aguarda la prestación del servicio.
Pueden ser usados como fuente de ventaja competitiva desde la óptica que facilitan la conexión a sistemas de proveedores y clientes.	

Infraestructura de los Servicios Web XML

La infraestructura está formada por cuatro servicios principales que emplean estos estándares, mostrados a continuación en la figura 4.1: [MS01.1]

98. No empleo el término *transportar* para no entrar en conflicto con definiciones de protocolos de transporte de datos. En un sentido estricto, XML no transporta datos por sí mismo, sino que les provee un formato para su transporte.

Figura 4.1 Infraestructura de los servicios Web XML

Directorio de Servicios Web XML

Como se explicó anteriormente, los datos XML pueden ser accedidos desde varios niveles, entre ellos el nivel formado por las aplicaciones. Lógicamente, el hecho de utilizar un servicio Web XML ya desarrollado requiere primero poder hallarlo. Sin un método de adecuado de localización, la búsqueda de éstos en Internet puede ser muy dificultosa. Los servicios de directorio para XML Web Services proveen un mecanismo de localización centralizado, sobre Internet, para encontrar descripciones de servicios Web que realizaron otras organizaciones. Un ejemplo de directorios de este tipo es *UDDI* (Universal Discovery, Description and Integration) de propiedad de la empresa Microsoft.

Descubrimiento de Servicios Web XML

El proceso de descubrimiento consiste en encontrar la ubicación de servicios Web específicos y la documentación que los describen. El proceso se lleva a cabo leyendo un documento de descubrimiento escrito en XML que contiene la ubicación de los documentos de descripción de los servicios.

Descripción de Servicios Web XML

El lenguaje *WSDL* (Web Services Description Language) se utiliza para escribir en XML una descripción formal de un servicio Web. Se crea un archivo *.wsdl* que define las interfaces de las funciones que ofrece el servicio y define el formato de los mensajes que el servicio entiende. WSDL es uno de los estándares de soporte creados por el World Wide Web Consortium.

Formatos de Servicios Web XML

Los servicios Web XML pueden construirse para trabajar con cualquier protocolo. No obstante, resulta más práctico desarrollarlos para que trabajen con tres protocolos abiertos de transportes muy frecuentemente utilizados:

- SOAP, Simple Object Access Protocol. Define cómo usar XML para representar datos y describir reglas de codificación, intercambio de mensajes y protocolos de transporte.
- HTTP.GET.
- HTTP.POST.

Protocolos de soporte para servicios Web

El World Wide Web Consortium[99] (W3C) ha desarrollado una serie de estándares para dar soporte al uso de los servicios Web XML.

WSDL[100]

Para que las aplicaciones puedan hacer uso de servicios Web sobre redes de datos es necesario contar con una forma estructurada de describir las comunicaciones de red. Esto sea hace posible a medida que se estandarizan los protocolos de comunicaciones y los formatos de los mensaje que éstos envían. WSDL es una forma de llevar a cabo la descripción de las comunicaciones mediante una gramática basada en XML. Entiende a las comunicaciones como un conjunto de puntos de enlace o *endpoints* capaces de intercambiar mensajes.

99. El World Wide Web Consortium (www.w3c.org) es una entidad creada en 1994 para tratar de aprovechar al máximo las características que ofrece la WWW. Su misión es: lograr el acceso universal de la Web, mantener un ambiente de desarrollo para que los usuarios de la Web hagan un uso óptimo de los recursos disponibles y guiar el desarrollo Web teniendo en cuenta aspectos legales, comerciales y sociales.
100. Referencia [W3C02]

Web Services Description Language (*WSDL*) significa lenguaje para la descripción de servicios Web. Es un lenguaje XML para la descripción de servicios de red definidos como un conjunto de endpoints y mensajes. Las operaciones y mensajes se describen de forma abstracta y luego se ligan a protocolos de red y a formatos de mensajes específicos para describir un endpoint. Los endpoints *concretos* relacionados se agrupan para formar endpoints *abstractos*, los **servicios**. La separación de endpoints concretos y abstractos permite la reutilización de las definiciones abstractas, los **mensajes**, que son descripciones de alto nivel de los datos a ser intercambiados.

WSDL permite describir por completo a los servicios Web XML, describe:

- Los métodos disponibles
- las distintas formas de invocación de los métodos
- los tipos de datos empleados por los métodos
- los formatos de mensajes de solicitud/respuesta enviados desde varios protocolos (SOAP, HTTP, etc.)

El desarrollador de aplicaciones móviles con Compact Framework no requiere conocer los detalles internos de funcionamiento de este lenguaje. Si se desarrolla una aplicación con servicios Web utilizando la herramienta Visual Studio .NET, éste crea automáticamente el código WSDL necesario.

SOAP [101]

Simple Object Access Protocol (*SOAP*), significa protocolo de acceso a objetos simples. Su función es proveer un mecanismo para intercambiar información estructurada y tipificada entre dos puntos de conexión, en un entorno distribuido utilizando XML.

Define una manera simple de expresar la semántica de una aplicación por medio de un modelo de empaquetado de módulos, y provee una forma de codificar la información contenida en los módulos. La especificación SOAP permite definir cómo usar XML para representar datos, describir reglas de codificación, esquemas de intercambio de mensajes y protocolos de transporte. La consecuencia inmediata de esto es la posibilidad de enviar información compleja a través de HTTP. Se entiende por información compleja a objetos formados por métodos y atributos. SOAP define entre otros: [SUN02.0]

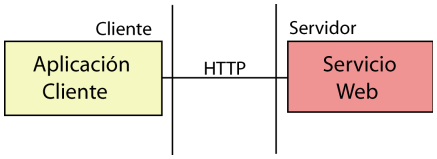
1. Un sobre SOAP. Indica **qué** contiene el mensaje, **quién** debe recibirlo y **cuándo** es obligatorio u opcional.
2. Reglas de codificación SOAP, que definen el mecanismo de serialización utilizado para intercambiar las instancias de los tipos de datos definidos por las aplicaciones.
3. Una representación SOAP de RPC, es decir, define una convención que puede ser usada para representar invocaciones a procedimientos remotos y sus respectivas respuestas. Para ser empleado como mecanismo de petición-respuesta, SOAP debe seguir convenciones de codificación y de estructura; esto se debe a que SOAP es naturalmente un mecanismo de envío en un solo sentido (emisor a receptor). Para invocar un procedimiento remoto se necesita la siguiente información: la dirección del nodo objetivo, el nombre del procedimiento en sí, identificación y valores de los argumentos a pasar al procedimiento e información para el protocolo de transporte subyacente a fin de informarle si es petición (GET) o respuesta (ej: POST).

Los detalles internos de los mensajes SOAP se abstraen en una clase *proxy* que se explica posteriormente en la sección de «proceso de desarrollo de una aplicación basada en XML Web services». Esta clase puede generarse automáticamente con la herramienta Visual Studio .NET, el cual además genera código que permite que un servicio Web XML utilice SOAP para comunicarse vía HTTP. El Compact Framework incluye clases para realizar implementaciones basadas en SOAP.

Proceso de Desarrollo de una Aplicación basada en XML Web Services

Las aplicaciones que hacen uso de servicios Web XML funcionan bajo un esquema de cliente/servidor; una aplicación expone funciones mediante un servicio Web y otra aplicación solicita los servicios ofrecidos. La figura 4.2 muestra a continuación el esquema de funcionamiento de un servicio Web:

Figura 4.2 Esquema Básico de un servicio Web



Para desarrollar una aplicación **cliente** de servicios Web XML se definen las siguientes fases:

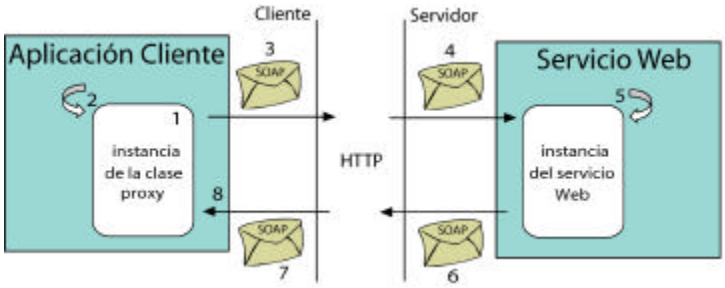
101. Referencia [W3C03]

Fase	Descripción
Directorio	Se necesita obtener un enlace al documento de descubrimiento del servicio Web deseado. Para utilizar un servicio Web propio[102], no es necesario realizar esta fase.
Descubrimiento	Se necesita obtener un enlace al documento de descripción escrito en WSDL, para aquellos servicios que no son propios.
Descripción	Se emplea el enlace (referencia Web) al documento de descripción para obtener el documento WSDL en cuestión.
Desarrollo	Con el documento WSDL, debe construirse una clase "*proxy*". Este proceso puede realizarse automáticamente con herramientas de software. La clase *proxy* representa el servicio Web en el cliente, permite el desarrollo de una aplicación que haga uso del servicio.
Ejecución[103]	La aplicación cliente invoca a los métodos del servicio Web XML, y recibe una respuesta por parte de éste. Esta fase se explica a continuación.

Fase de Ejecución de un Servicio Web XML

La ejecución de un servicio Web XML comienza a partir de que el cliente instancia la clase *proxy* y finaliza cuando recibe los resultados esperados. Para comprender este proceso, es necesario visualizarlo como una serie de ocho pasos sucesivos. La figura 4.3 sirve de apoyo para la explicación que se provee a continuación: [MS01.1]

Figura 4.3 Ejecución de un servicio Web XML



1. La aplicación del cliente crea una instancia de la clase *proxy*
2. El cliente invoca a algún método de la clase
3. Los argumentos del método invocado son serializados[104] en un mensaje *SOAP* y este mensaje se envía a través de la conexión de red al servicio Web.
4. El servidor recibe el mensaje *SOAP* y crea una instancia localmente de la clase que implementa al servicio Web. Toma el objeto serializado y hace el proceso inverso para obtener los parámetros del mensaje *SOAP*.
5. La aplicación en el servidor invoca al método indicado con los parámetros que recibió. El método se ejecuta y se obtienen parámetros de salida y un valor de retorno.
6. La aplicación en el servidor crea un mensaje *SOAP* e incluye los parámetros de salida y valor de retorno en él; el mensaje se envía a través de la conexión de red.
7. La aplicación cliente recibe el mensaje *SOAP* y extrae la información contenida en él.
8. El cliente obtiene los resultados de su solicitud de servicio.

102. Con «*propio*» estoy indicando que no es un servicio Web desarrollado por terceras partes sino por el mismo desarrollador de la aplicación que lo utiliza.
103. Esta fase no se halla realmente en el proceso de desarrollo como tal de una aplicación cliente pero se incluye para dar continuidad al proceso.
104. La serialización es una función que ofrece .NET que permite «empaquetar» un objeto instanciado a un formato que puede ser binario para su envío en un flujo de datos determinado, como una conexión de red, un archivo, etc.

El proceso anteriormente explicado vale para el Framework estándar y para el Compact Framework, pero existe una diferencia en el paso 3: el Compact Framework no provee soporte para la serialización binaria (mediante las clases *BinaryFormatter* o *SoapFormater)*. No obstante existe una solución alternativa: la infraestructura de los servicios Web XML provee soporte para la transmisión de objetos de datos utilizando *SOAP*.

Consideraciones Previas de Diseño en Compact Framework

En el capítulo 2 se explicaron las consideraciones de eficiencia que deben tenerse presente para el desarrollo de aplicaciones en dispositivos móviles usando el Compact Framework. En las secciones anteriores hemos visto las características principales del uso de servicios Web basados en XML; a continuación se exponen aspectos previos a tener en cuenta para implementar los servicios usando el Compact Framework.

Referencias Web

El servidor Web no se encuentra localmente en el dispositivo, por lo tanto las referencias Web no pueden realizarse usando *localhost*[105].

Tamaño de los DataSets

Los DataSets son contenedores *offline* de información. En capítulos anteriores hemos visto que una decisión de diseño válida es no contar con una base de datos local en un dispositivo móvil, sino, establecer una conexión de red y solicitar subconjuntos de datos bajo demanda. El uso de servicios Web es muy adecuado para llevar a cabo estas operaciones. No obstante, hay que evaluar previamente la magnitud de datos a transmitir por un método de un servicio Web. Se recomienda enviar pequeñas cantidades de información (idealmente pocos registros) por cada envío, ya que el cliente es un dispositivo móvil y su capacidad de procesamiento es muy inferior respecto del servidor.

DataSets con Tipo

Los DataSets con tipo no están soportados por el Compact Framework. Por esto, no deben emplearse en los argumentos ni como tipo de retorno de los métodos del servicio Web. Esto puede resultar confuso ya que un servicio Web se ejecuta en el servidor que corre el .NET Framework estándar donde sí están soportados, pero su contraparte en el cliente no puede tener acceso a estos tipos de datos. Usando DataSets sin tipo (*untyped* DataSets) se soluciona esta limitación.

Para aclarar los conceptos al lector, los DataSet (ya expuestos en esta tesina) se presentan en dos versiones, con y sin tipo. Los primeros permiten definir un DataSet en tiempo de diseño ya que se conoce el esquema[106] previamente. El esquema resultante es de tipo seguro, pues todos los tipos han sido definidos antes de la ejecución y se conoce de antemano qué conversiones de datos se realizarán (por asignación). Los DataSet sin tipo se emplean generalmente cuando no se conoce en tiempo de diseño el esquema de los datos, con lo cual el entorno genera el DataSet en tiempo de ejecución; si no es manejado correctamente puede provocar errores de tipo durante la ejecución.

Operaciones Asincrónicas

En la tabla 4.1 de este mismo capítulo se mencionó que la solicitud un servicio puede requerir un lapso de tiempo prologando en completarse, dependiendo de la complejidad del servicio Web. Una forma de prever este hecho a nivel de programa es invocar a servicios Web de manera asincrónica[107]. Así, un thread aparte mantiene una solicitud de servicio y otras tareas pueden ser realizadas mientras la solicitud se completa. El usuario puede ser informado por la interfaz gráfica mediante algún otro control de interfaz.

5.1 Introducción

Esta sección final tiene como objetivo integrar en un escenario simplificado algunos conceptos fundamentales que se han expuesto a lo largo de esta tesina; para ello realizo un desarrollo.

Se presenta el escenario compuesto por tres piezas de software: dos aplicaciones y un servicio Web para una universidad. La universidad cuenta con un **módulo de administración** de cursos extracurriculares que los alumnos deben realizar para cumplir con un requerimiento mínimo de créditos por semestre. Este módulo ofrece funciones básicas de administración de cursos, consultas de inscripciones y asistencias, y

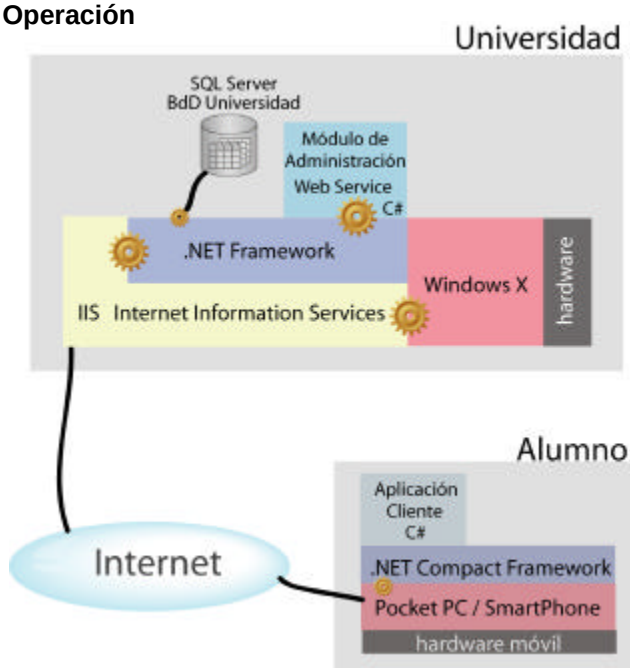
105. Localhost es un nombre que emplea IIS (Internet Information Services) para que las aplicaciones Web puedan ejecutarse localmente haciendo uso de un navegador, es decir, como si se tratara de una aplicación remota.
106. El esquema es la representación de la estructura de los datos. Visual Studio almacena el esquema en formato XML.
107. En este contexto, la sincronía implica secuencialidad: una tarea debe completarse para que otra comience a ejecutarse.

es operado por el personal de la universidad. En un servidor de la universidad está publicado un **servicio Web** que ofrece a los alumnos la posibilidad de consultas e inscripción a los cursos disponibles, junto a otras funciones que se ampliarán más adelante. Por otra parte, la universidad ha desarrollado una **aplicación móvil** que será provista a los alumnos para que instalen en sus dispositivos móviles personales. La aplicación cliente permite a los alumnos consultar los cursos disponibles, inscribirse y cancelar inscripciones haciendo uso del servicio Web.

5.2 Ambiente y Modo de Operación

Un ambiente típico de operación para el escenario de la universidad y los cliente móviles utilizados por los alumnos puede ser un equipo servidor corriendo Windows[108] como sistema operativo, con Internet Information Services[109] instalado y configurado, el .NET Framework y SQL Server. El lector puede ver en la figura 5.1 este ambiente de operación.

Figura 5.1 Ambiente de Operación



No realizo conjeturas respecto de la distribución física de cada componente de software, es decir, podría optarse por situar la base de datos en un equipo aparte del servidor Web y poner el módulo de administración de cursos en un tercer equipo o simplemente situar todo junto. En favor de ilustrar más claramente el escenario[110] vamos a suponer que todos los componentes de software de la universidad se ejecutan sobre un equipo único al cual llamaremos «servidor».

En el servidor se encuentra la base de datos denominada «*Universidad*» que contiene las tablas, datos y procedimientos para la gestión de los cursos, inscripciones y asistencias. La base de datos fue creada en el sistema de gestión de bases de datos SQL Server, el cual provee los métodos de acceso, actualización y asegura condiciones de integridad de datos[111]. Por otro lado, cada alumno posee un dispositivo móvil con algún sistema operativo que soporte el Compact Framework (como Pocket PC o Smartphone), el Compact Framework instalado para ejecutar la aplicación cliente que desarrolló la universidad y la aplicación cliente instalada.

La aplicación de administración de cursos trabaja sobre la base *Universidad* a través del marco de trabajo[112]. Todas las consultas se harán sobre SQL Server y los cambios generados desde la aplicación se reflejan de forma permanente en la base.

El servicio Web que la universidad pone a disposición de los alumnos reside en el servidor. Fue escrito en C# y al igual que el módulo de administración trabaja sobre la base *Universidad*. El servicio Web es una

108. En cualquiera de sus versiones que soporte el uso de Internet Information Services, el .NET Framework y SQL Server.
109. Ver glosario para más detalle.
110. y en desmedro de las buenas prácticas de diseño de sistemas y seguridad (las cuales no son objeto de esta aplicación de tesina).
111. Me refiero al principio de integridad referencial que asegura que si una tupla en una relación B hace referencia a otra relación A, entonces la tupla debe existir en la última relación. Ningún valor de clave primaria puede ser nulo.
112. Haciendo uso de las características de ADO.NET: un conjunto de objetos que proveen acceso a datos.

pieza de software aparte y se ubica bajo el ámbito de una «aplicación IIS[113]». Es decir, a partir de que IIS adquiere el recurso «servicio Web», la ejecución del mismo pasa a estar en sus manos. Los servicios Web ofrecidos que requieren interacción con otros recursos (como la base *Universidad)* lo harán por medio del .NET Framework.

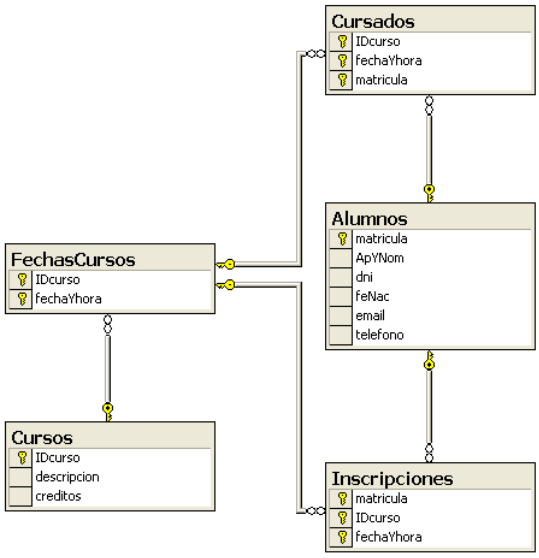
Desde la perspectiva del cliente móvil, el alumno ejecuta su aplicación para tener acceso a la información de los cursos. Para ello, lo único que se requiere es una conexión a Internet. La aplicación se ejecuta sobre el .NET Compact Framework y cada vez que el alumno solicita un servicio Web, el servidor (con IIS) atiende su petición enviando o recibiendo datos en formato XML. Los detalles sobre los servicios en concreto que se ofrecen a los alumnos se cubrirán posteriormente en este capítulo.

5.3 Algunos detalles de implementación

Base de datos *Universidad*

La base está compuesta por 5 tablas, denominadas Alumnos, Cursos, FechasCursos, Inscripciones y Cursados. La figura 5.2 muestra el diagrama de la base.

Figura 5.2 Diagrama de la base Universidad



Cada caja representa una de las tablas indicando su nombre en la parte superior y contiene una lista en su interior de los atributos que dispone. Los atributos que están precedidos por un icono con una llave amarilla forman parte de la clave primaria de la tabla. Las líneas que conectan dos tablas indican relación de clave externa (o foránea).

Tabla Alumnos

Esta tabla registra los datos personales de los alumnos.

Los alumnos se identifican unívocamente a través del atributo ***matrícula***. Se registra adicionalmente el nombre completo, indicando primero el/los apellido/s seguido del nombre de pila, el número de DNI o documento correspondiente, la fecha de nacimiento, una dirección de correo electrónico de contacto y un teléfono de contacto. Ningún atributo admite valores nulos.

Tabla Cursos

Esta tabla registra los cursos extracurriculares que la universidad ofrece a los alumnos.

Los cursos se identifican de manera unívoca a través del atributo ***IDcurso***. Se registra un texto de descripción para el curso, el número de créditos que otorga a los alumnos en concepto de asistencia, y la fecha y hora en que se dicta el curso. Ningún atributo admite valores nulos.

Tabla FechasCursos

Esta tabla registra las distintas fechas en que se pueden dictar los cursos. Un mismo curso puede ser dictado en varias oportunidades.

113. Para IIS, una aplicación es cualquier archivo ejecutado dentro de un conjunto definido de directorios pertenencientes a un sitio Web
. Cada sitio Web es administrado de manera independiente por IIS.

Tabla Inscripciones

Esta tabla registra las inscripciones de los alumnos a los cursos. Un alumno puede inscribirse a todos los cursos que desee.

Tabla Cursados

Esta tabla registra los alumnos que asistieron a cursos, junto al curso realizado. Se registra la matrícula del alumno, el curso que tomó y la fecha en que asistió al curso.

Módulo de Administración de Cursos

Esta aplicación es operada exclusivamente por el personal de la universidad. Las funciones que ofrece son:

1. Consultas de alumnos existentes en la base
2. ABMC[114] de cursos
3. ABMC de fechas de cursos
4. Consultas de inscripciones de alumnos a cursos
5. Consultas de cursos realizados por un alumno determinado
6. Consultas de concurrencia de alumnos a un curso determinado
7. Confirmación de asistencia de un alumno a un curso
8. Registración de ausencia de un alumno a un curso

Este módulo es una aplicación .NET implementada en C# y por lo tanto corre sobre el Framework completo.

Servicio Web a Alumnos

Partamos de la base que un servicio Web se compone de una serie de métodos o procedimientos que se exponen al cliente, donde cada procedimiento tiene una signatura[115] y un tipo de retorno.

Los servicios a los que el alumno podrá acceder mediante su dispositivo personal se establecen mediante los métodos definidos en la clase que implementa el servicio Web. Tales métodos pueden visualizarse en la tabla 5.1.

Tabla 5.1 Métodos del servicio Web

Tipo de Retorno	Método	Lista de Parámetros
DataSet	EnviarCursosDisponibles	*sin parámetros*
DataSet	ConsultarCursosRealizados	matrícula
bool	InscribirAlumno	matrícula, curso, fecha
bool	CancelarInscripcionAlumno	matrícula, curso, fecha
DataSet	ConsultasCursosInscriptos	matrícula

EnviarCursosDisponibles

Cuando el cliente invoca este método recibe una lista de los cursos disponibles a los cuales puede inscribirse. Se indica tema del curso, fecha, hora y créditos que se otorgan. La lista se obtiene a través de un objeto DataSet.

ConsultarCursosRealizados

Este método requiere como parámetro la matrícula del alumno y retorna una lista con los cursos que el alumno ha realizado, es decir, aquellos cursos a los que el alumno se inscribió anteriormente y asistió. Se

InscribirAlumno

Este método recibe como parámetros la matrícula del alumno, un curso determinado, fecha y hora en que se dicta. Permite a un alumno realizar la inscripción a ese curso mediante su dispositivo. Retorna un valor lógico de verdad indicando si la inscripción ha sido satisfactoria.[116]

114. ABMC es un acrónimo para una interfaz de altas, bajas, modificaciones y consultas.
115. Se entiende por signatura a un nombre de método, seguido de una lista de parámetros con tipo.
116. Podría ocurrir que el alumno ya se encuentre inscripto al curso en esa misma fecha o alguna condición excepcional como indisponibilidad temporal de la base de datos.

CancelarInscripcionAlumno

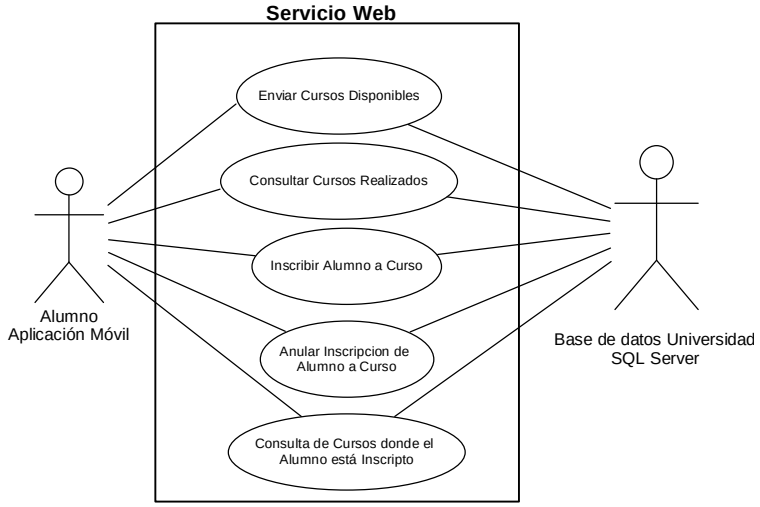
Este método recibe como parámetros la matrícula del alumno, un curso determinado y la fecha en que se dicta. Permite a un alumno cancelar una inscripción realizada previamente al curso especificado. Retorna un valor lógico de verdad indicando si la cancelación ha sido satisfactoria.

ConsultasCursosInscriptos

Permite obtener, mediante un DataSet, una lista de los cursos a los que el alumno se ha inscripto, indicando tema del curso, fecha, hora y créditos que se otorgan. Requiere como parámetro la matrícula del alumno.

La figura 5.3 a continuación muestra un diagrama de casos de uso a fin de que el lector identifique rápidamente y de forma visual las opciones ofrecidas al alumno.

Figura 5.3. Funciones ofrecidas al alumno



5.4 Solicitud de un servicio Web

En tiempo de diseño, se agrega en la aplicación cliente una referencia Web[117] al servicio Web publicado por la universidad. A través de la referencia, el software cliente accede a la información de descripción de los servicios Web. La descripción del servicio Web se escribe en *WSDL* y contiene la información necesaria para establecer el intercambio de datos. Así, el cliente puede conocer qué métodos hay disponibles en el servicio, las distintas formas de invocación, los tipos de datos que se utilizan, los formatos de mensajes, etc. Con esta información, el entorno de desarrollo genera una clase *proxy* para uso en el cliente. La clase *proxy* es la que expone a la aplicación móvil los métodos mostrados en la tabla 5.1.

La clase *proxy*, además, representa la abstracción necesaria para el manejo de mensajes *SOAP*[118], el protocolo que se ocupa del envío y recepción de mensajes para llevar a cabo las invocaciones a procedimientos. *SOAP* es empleado precisamente para realizar las solicitudes de invocación a procedimientos remotos, y consecuentemente para el retorno de los resultados.

Los mensajes son enviados a través de *HTTP*, un protocolo Web estándar. Este protocolo cuenta con dos acciones principales que son GET y POST, empleados para codificar y pasar parámetros de una solicitud. GET y POST permiten el paso de parámetros bajo la forma de pares *(nombre, valor)* que son adosados a la *URL* de la dirección solicitada.

Por otro lado, *SOAP* especifica cómo utilizar XML para representar los datos, para describir las reglas de codificación de los datos y los mecanismos de intercambio con los protocolos de transporte. Con esto se logra una forma de enviar datos más complejos a través de HTTP y permite así representar esos datos como un objeto con propiedades y métodos en una aplicación cliente. [MS01.1]

Situándonos de nuevo en el desarrollo de la universidad, vamos a hacer un seguimiento de lo que ocurre cuando un alumno desea usar algunas de las funciones que su aplicación móvil le ofrece. Supongamos que el alumno desea conocer qué cursos tienen fechas programadas para dictarse en los próximos días. El alumno inicia la aplicación móvil que la universidad le proveyó, y elige la opción de obtención de cursos disponibles. Se desencadenan entonces una serie de sucesos que expongo a continuación:

✍ Se crea una instancia de la clase *proxy* del servicio Web

117. Una referencia Web es una dirección de la cual puede obtenerse el archivo de descripción del servicio Web.
118. Protocolo expuesto anteriormente en el capítulo 4.

- La aplicación móvil informa a la clase *proxy* que se desea invocar al método *EnviarCursosDisponibles()*
- La clase *proxy* crea un mensaje *SOAP* que contiene una solicitud de invocación al método mencionado, e incluye los parámetros de llamada correspondientes. En este caso, el método a invocar no recibe parámetros
- Entre la aplicación cliente y el servidor se generan una serie de acciones HTTP.GET y HTTP.POST para el transporte del mensaje *SOAP* a través de Internet. En el servidor, IIS se encarga de procesar las solicitudes Web.
- El servicio Web situado en el servidor recibe el mensaje *SOAP* que envió el cliente y crea en memoria una instancia de la clase que implementa el servicio. Se extrae la información del mensaje *SOAP* y se obtienen los parámetros del método a invocar.
- Se ejecuta en el servidor el método *EnviarCursosDisponibles()*: el Framework conecta a la infraestructura del servicio Web con la base *Universidad* y realiza la consulta de los cursos disponibles. Como el método debe retornar un *DataSet*, la infraestructura arma este contenedor de información con el resultado de la consulta.
- Un nuevo mensaje *SOAP* es creado en el servidor conteniendo la información de retorno serializada[119]. Se representa la información usando XML. La siguiente sección muestra el resultado de la representación XML de los datos.
- Entre la aplicación cliente y el servidor se generan nuevamente una serie de acciones HTTP.GET y HTTP.POST para el transporte del mensaje *SOAP* a través de Internet.
- La clase *proxy* en el cliente recibe el mensaje *SOAP*, y extrae la información contenida. Se deserializa la información XML recibida y se genera localmente un nuevo objeto *DataSet*.
- La aplicación .NET en el cliente obtiene el *DataSet* y lo muestra en pantalla mediante algún control de usuario[120].

¿Cómo se representa el *DataSet* del ejemplo mediante XML?

Para poder ilustrar cómo el marco de trabajo genera una representación XML de la información contenida en un *DataSet*, vamos a suponer el siguiente juego de datos contenido en las tablas *Cursos* y *FechasCursos*:

Tabla 5.2 Cursos

IDcurso	descripcion	creditos
1	Logica Borrosa	5
2	Matematica Financiera	4
4	Compiladores e Interpretes	6
5	Programacion en Logica	4
6	Idioma Aleman	4
10	Idioma Portugues	5

Tabla 5.3 Fechas Cursos

IDcurso	fechaYhora
1	10/08/2004
2	10/08/2004
4	10/08/2004
4	27/08/2004
4	21/10/2004
5	01/04/2004
5	03/08/2004
5	25/08/2004
5	27/08/2004
6	10/05/2004
6	20/05/2004
6	25/05/2004

119. Para más detalle, ver el glosario al final de la tesina.
120. Por ejemplo, puede emplease un DataGrid para mostrar los cursos disponibles (control ya expuesto en el capítulo 3).

Siguiendo la línea de ejemplo de la sección anterior, cuando se ejecuta el método *EnviarCursosDisponibles()* se obtiene el conjunto de datos que se muestra a continuación:

```
 IDcurso     descripcion                  fechaYhora             creditos
----------- ---------------------------- ---------------------- -----------
1           Logica Borrosa               10/08/2004             5
2           Matematica Financiera        10/08/2004             4
4           Compiladores e Interpretes   10/08/2004             6
4           Compiladores e Interpretes   27/08/2004             6
4           Compiladores e Interpretes   21/10/2004             6
5           Programacion en Logica       01/04/2004             4
5           Programacion en Logica       03/08/2004             4
5           Programacion en Logica       25/08/2004             4
5           Programacion en Logica       27/08/2004             4
6           Idioma Aleman                10/05/2004             4
6           Idioma Aleman                20/05/2004             4
6           Idioma Aleman                25/05/2004             4
```

No obstante, este resultado devuelto por la consulta a SQL Server debe ser debidamente adaptado en formato y enviado al cliente en el *DataSet*. Este contenedor define internamente un elemento *DataTable*[121] formado por *DataColumns*[122] para cada atributo y se crea un *DataRow*[123] para cada uno de los registros devueltos por la consulta. Hasta este punto podemos decir que se ha generado un «esquema» del contenedor y que ha sido rellenado con los datos de la consulta. Ahora es cuando el contenedor es representado usando XML.

La representación XML del *DataSet* se divide en dos secciones, la primera define el esquema del contenedor, y la segunda contiene los datos.

Cuando la clase *proxy* en el cliente extrae la información XML es capaz de crear en memoria un nuevo objeto *DataSet* basándose en ella. Esto es posible puesto que el Compact Framework comparte un subconjunto de la librería de clases del Framework estándar y por lo tanto entiende la definición XML de un objeto *DataSet*.

A continuación puede verse el código XML generado a partir del contenedor y ésta información es la que se transporta en definitiva a través de la Web:

```
<?xml version=»1.0" encoding=»utf-8"?>
<DataSet xmlns=»http://tempuri.org/»>
   <xs:schema id=»Cursos» xmlns=»» xmlns:xs=»http://www.w3.org/2001/XMLSchema»
xmlns:msdata=»urn:schemas-microsoft-com:xml-msdata»>
      <xs:element name=»Cursos» msdata:IsDataSet=»true» msdata:Locale=»es-AR»>
        <xs:complexType>
           <xs:choice maxOccurs=»unbounded»>
            <xs:element name=»Table»>
              <xs:complexType>
                <xs:sequence>
                     <xs:element name=»IDcurso» type=»xs:int» minOccurs=»0" />
                      <xs:element name=»descripcion» type=»xs:string» minOccurs=»0" />
                      <xs:element name=»fechaYhora» type=»xs:dateTime» minOccurs=»0" />
                     <xs:element name=»creditos» type=»xs:int» minOccurs=»0" />
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
   </xs:schema>
   <diffgr:diffgram xmlns:msdata=»urn:schemas-microsoft-com:xml-msdata»
xmlns:diffgr=»urn:schemas-microsoft-com:xml-diffgram-v1">
     <Cursos xmlns=»»>
        <Table diffgr:id=»Table1" msdata:rowOrder=»0">
          <IDcurso>1</IDcurso>
           <descripcion>Logica Borrosa</descripcion>
           <fechaYhora>2004-08-10T00:00:00.0000000-03:00</fechaYhora>
          <creditos>5</creditos>
       </Table>
        <Table diffgr:id=»Table2" msdata:rowOrder=»1">
          <IDcurso>2</IDcurso>
```

121. La clase que representa una tabla de datos.
122. La clase que representa cada atributo de datos que posee una tabla.
123. La clase que representa cada registro que contiene un DataTable.

```
            <descripcion>Matematica Financiera</descripcion>
            <fechaYhora>2004-08-10T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table3" msdata:rowOrder=»2">
          <IDcurso>4</IDcurso>
            <descripcion>Compiladores e Interpretes</descripcion>
            <fechaYhora>2004-08-10T00:00:00.0000000-03:00</fechaYhora>
          <creditos>6</creditos>
        </Table>
          <Table diffgr:id=»Table4" msdata:rowOrder=»3">
          <IDcurso>4</IDcurso>
            <descripcion>Compiladores e Interpretes</descripcion>
            <fechaYhora>2004-08-27T00:00:00.0000000-03:00</fechaYhora>
          <creditos>6</creditos>
        </Table>
          <Table diffgr:id=»Table5" msdata:rowOrder=»4">
          <IDcurso>4</IDcurso>
            <descripcion>Compiladores e Interpretes</descripcion>
            <fechaYhora>2004-10-21T00:00:00.0000000-03:00</fechaYhora>
          <creditos>6</creditos>
        </Table>
          <Table diffgr:id=»Table6" msdata:rowOrder=»5">
          <IDcurso>5</IDcurso>
            <descripcion>Programacion en Logica</descripcion>
            <fechaYhora>2004-04-01T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table7" msdata:rowOrder=»6">
          <IDcurso>5</IDcurso>
            <descripcion>Programacion en Logica</descripcion>
            <fechaYhora>2004-08-03T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table8" msdata:rowOrder=»7">
          <IDcurso>5</IDcurso>
            <descripcion>Programacion en Logica</descripcion>
            <fechaYhora>2004-08-25T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table9" msdata:rowOrder=»8">
          <IDcurso>5</IDcurso>
            <descripcion>Programacion en Logica</descripcion>
            <fechaYhora>2004-08-27T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table10" msdata:rowOrder=»9">
          <IDcurso>6</IDcurso>
            <descripcion>Idioma Aleman</descripcion>
            <fechaYhora>2004-05-10T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table11" msdata:rowOrder=»10">
          <IDcurso>6</IDcurso>
            <descripcion>Idioma Aleman</descripcion>
            <fechaYhora>2004-05-20T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
          <Table diffgr:id=»Table12" msdata:rowOrder=»11">
          <IDcurso>6</IDcurso>
            <descripcion>Idioma Aleman</descripcion>
            <fechaYhora>2004-05-25T00:00:00.0000000-03:00</fechaYhora>
          <creditos>4</creditos>
        </Table>
      </Cursos>
    </diffgr:diffgram>
</DataSet>
```

Como se anticipó anteriormente, se puede observar en el código XML dos secciones distintas. La primera sección, coloreada en azul a fines ilustrativos, define el esquema del contenedor de datos. Entre otras cosas puede verse la definición de elementos con su nombre y tipo, por ejemplo:

<xs:element name=»IDcurso» type=»xs:int»/> define al elemento IDcurso de tipo int (entero).

La segunda sección, señalada en color negro, representa los datos contenidos en el esquema. Se muestran en color rojo los datos en si.

El desarrollo de aplicaciones móviles plantea un área de creciente importancia para las organizaciones y los profesionales de la tecnología informática. Tal como se expuso en el capítulo 1, empresas consultoras como Gartner registran un crecimiento del uso de dispositivos móviles en los últimos años y proyectan al corto plazo un importante incremento adicional. Según los estudios de estas consultoras, la utilización de los dispositivos móviles dentro de las organizaciones es una fuente potencial de ahorro de dinero y de oportunidades comerciales. Una meta principal de la incorporación de tecnologías móviles en la empresa es la extensión de los sistemas existentes.

Para incorporar tecnologías móviles en una empresa de manera eficaz, es conveniente que el personal de IT[124] realice un análisis de los riesgos potenciales que esta tecnología pueda representar, tanto sobre la economía, sobre la integración con los sistemas existentes, la seguridad y el impacto sobre los recursos humanos.

Se debe tener en cuenta que el desarrollo de software para dispositivos móviles requiere de algunas técnicas de especial consideración debido a las restricciones de hardware y software que poseen los dispositivos actuales. Entre esas técnicas destaco aquellas ligadas al uso eficiente de los recursos de memoria de ejecución, de almacenamiento, uso del espacio de visualización y métodos de entrada de datos.

Es importante contar con un ambiente de desarrollo formado por herramientas y lenguajes que ofrezcan a los desarrolladores facilidad de aprendizaje y la posibilidad de generar código administrado[125], debido a las exigencias de los mercados de lograr desarrollos rápidos, confiables y que puedan ser liberados en el menor tiempo posible.

El marco de trabajo empleado en esta tesina es el .NET Compact, una versión del Framework completo para desarrollos móviles. Una consecuencia directa de esto es que, si una empresa cuenta con desarrolladores con experiencia en tecnologías y lenguajes de .NET, entonces no resulta necesario que éstos aprendan nuevos lenguajes y entornos de desarrollo. Si bien .NET puede ser empleado para aplicaciones móviles, se requiere en los implementadores un cambio en la forma de programar y un uso muy medido de los recursos.

No obstante, los constantes avances en la industria electrónica van a ir minimizando las restricciones de memoria y de performance de los equipos móviles, con lo cual presupongo que se simplificarán más estas cuestiones de «gestionar las limitaciones» al tiempo que se introducirán nuevos desafíos (tal vez relacionados a otras áreas como la seguridad).

De mi experiencia en el uso del marco compacto de trabajo para el desarrollo de la aplicación de la tesina surge que: el marco ofrece características de escritura de código portable entre los diversos tipos de dispositivos que el mercado ofrece, pero existen casos en que el código debe sufrir adaptaciones. Esto se debe fundamentalmente a:

- los distintos tamaños del área de visualización, que implican diseños diferentes de interfaces gráficas
- las diferencias en las capacidades de almacenamiento. Un ejemplo típico es una aplicación que se desarrolla para Pocket PC y para Smartphone
- los medios para el ingreso de datos, pues hay dispositivos que tienen un pequeño teclado físico, otros posee un teclado emulado por software y otros un teclado alfanumérico como es el caso de los Smartphone.

La funcionalidad que tiene el marco compacto para el desarrollo de aplicaciones móviles, como la librería de clases (*Base Class Library*), ofrecen un soporte amplio para asistir al desarrollador en su tarea. El desarrollador sólo debe ocuparse de las cuestiones específicas de su aplicación y no de implementar artefactos de uso frecuente: los controles de usuario, manejo de estructuras de datos y funciones de acceso a datos[126] usuales son provistas por el entorno, sólo deben ser utilizadas adecuadamente.

Es importante remarcar que hasta el momento, el marco compacto se encuentra reducido respecto del marco estándar y esto hace que sea necesario revisar el uso de las clases de la librería si el desarrollador proviene de la plataforma .NET estándar. Puede ocurrir que un desarrollador acostumbrado a utilizar objetos de la librería de clases no pueda utilizar determinados métodos o propiedades que no tienen soporte en el marco compacto por una cuestión de espacio de almacenamiento.

En mi experiencia[127] de uso de las características de acceso a datos, no he encontrado dificultades de implementación en el marco compacto, pues se realiza de la misma forma que en el marco estándar. Lo

124. Acrónimo en inglés de Information Technology, que significa tecnología de la información.
125. Para más información ver el glosario.
126. Por acceso a datos se entiende una aplicación que requiere del uso/manipulación de datos de una fuente externa a la aplicación. Como ejemplo, los accesos a datos en un servidor de base de datos SQL u Oracle.
127. Basándome en el desarrollo de la aplicación móvil y de la aplicación .NET estándar para el ejemplo de la «Universidad».

mismo sucedió, en general, con el manejo de formularios de tipo Windows y con el uso de los controles[128] de usuario. Mi experiencia previa en el lenguaje C# me habilitó para comenzar con el desarrollo en el marco compacto sin que fuera necesario incorporar un nuevo lenguaje para desarrollar la aplicación móvil.

Para finalizar, el manejo que el marco compacto hace de los servicios Web, hace que la consumición de servicios Web sea una tarea relativamente sencilla para el desarrollador. Esto se debe a que la complejidad subyacente de los servicios Web XML (donde intervienen múltiples protocolos) se reduce a invocar métodos a la manera tradicional.

Bibliografía

[DE97]
Libro
«Qué Será». Michael L. Dertouzos
1997. Editorial Planeta.
ISBN 0-060251479-9

[CP00]
Artículo
«What is a framework». Mark Clifton
The Code Project
http://www.codeproject.com/gen/design/WhatIsAFramework.asp

[FO04]
Sitio Web
«Free Online Dictionary of Computing»
2004. Reino Unido
http://foldoc.doc.ic.ac.uk

[GA01]
Material Apoyo de Conferencia
«Mobile and Wireless Computing: The Next User Revolution». Joe Baylock.
The 6th Annual Future of 'E'
Junio de 2001. Gartner

[GUE98]
Libro
«Programming Language Concepts» Ghezzi, Jazayeri.
1998. John Wiley & Sons, Inc.
ISBN 0-471-10426-4

[HK02.0]
Libro Web
Chapter4: «Devices and applications»
City University of Hong Kong, Eindhoven University of Technology (The Netherlands), University of Tilburg (The Netherlands), Grenoble Social Science University (Francia) & University of Central Florida (EUA).
2002. HKNet 5
http://hknet.tm.tue.nl/section41/lead.html

[IC01]
Presentación Electrónica
«Mobile Software Development». Brant Coghlan
2001. ICE, The Technology Conference. Canadá
http://www.iceconference.com/ICE2001/Program/Presentations/Brant%20Coghlan%20-%20Mobile%20SW%20Dev.ppt

128. Controles de usuario son aquellos elementos de la interfaz de usuario que permiten el ingreso/visualización de datos y la interacción con el operador de la aplicación.

[KO03.0]
Paper
«Evaluation of Mobile Applications - software-technical and human aspects»
Universidad de Koblenz-Landau (Alemania)
2003. Grupo MARC (Mobile Applications Research Center)
http://mobileapps.uni-koblenz.de/publications/onlinedoc/ecite02.pdf

[MS01.0]
Whitepaper
«Mobile Devices in the Enterprise». Douglas Dedo.
Octubre de 2001. Microsoft Corporation. Mobility Group
http://www.microsoft.com/windowsmobile/resources/whitepapers/devicesinenterprise.mspx

[MS01.1]
Libro
«Microsoft .NET Compact Framework» Andy Wigley & Stephen Wheelwright
2003. Microsoft Press
ISBN 0-7356-1725-2

[MS01.2]
Paper
«Mobile Enterprise Solutions: What Is the Appropriate Pocket-Size Platform?»
Douglas Dedo. Mobility Group
Octubre de 2001. Microsoft Corporation
http://www.microsoft.com/technet/itsolutions/mobile/evaluate/mobilwhy.mspx

[MS01.3]
Artículo
«Windows Mobile-Based Devices and Security: Protecting Sensitive Business Information»
Douglas Dedo. Mobility Group
Febrero de 2004. Microsoft Corporation
www.microsoft.com/windowsmobile/

[MS01.4]
Artículo
«XML Web Services Basics»
Roger Walter. Microsoft
Diciembre de 2001
www.msdn.microsoft.com

[MS01.5]
Artículo
«The Birth of XML Web Services»
Aaron Skonnard. Microsoft
Octubre de 2002
www.msdn.microsoft.com

[MS03.0]
Paper
«The Return On Your Mobility Investment. Enterprise Opportunities For Windows Mobile-Based Pocket Pcs and Smartphones»
Douglas Dedo. Mobility Group
Marzo de 2003. Microsoft Corporation
http://www.microsoft.com/windowsmobile/resources/whitepapers/mobilityROI.mspx

[MS03.1]
DVD
«Microsoft Mobility Developer Conference 2003»
Bill Gates y equipo.
2003. Microsoft Corporation

[MS03.2]
Artículo
«Comparison of Windows CE .NET 4.2, Pocket PC 2002 and Windows Mobile 2003 for Pocket PCs»
2003. Microsoft
www.msdn.microsoft.com/mobility

[MS03.3]
Artículo
«Differences in Microsoft Compact Framework Development between the Pocket PC and Windows CE .NET»
2003. Jim Wilson. Microsoft
www.msdn.microsoft.com/mobility

[MS03.4]
Artículo
«SQLServer CE Books Online. Hardware and Software Requirements».
Microsoft
www.msdn.microsoft.com

[MS03.5]
Artículo
«Smart Device Programmability for Embedded Applications»
Microsoft
http://msdn.microsoft.com/vstudio/device/smartdevprog.aspx

[MS03.6]
Artículo
«Introduction to Development Tools for Windows Mobile-based Pocket PCs and Smartphones»
Microsoft
www.msdn.microsoft.com

[SDK03.0]
«SmartPhone 2003 Software Developers Kit»
2003. Microsoft

[SDK03.1]
«Pocket PC Software Developers Kit»
2003. Microsoft

[SOM02]
Libro
«Ingeniería de Software. 6ta Edición» Ian Sommerville
2002. Adisson Wesley
ISBN 970-26-0206-9

[SUN02.0]
Artículo
«Overview of SOAP» Tom Clements
2002. Sun Microsystems
http://java.sun.com/developer/technicalArticles/xml/webservices/

[STA01]
Libro
«Operating Systems. 4th Ed» William Stallings
2001. Prentice Hall
ISBN 0-13-031999-6

[W3C01]
Página Web
«XML Tutorial»
W3 Schools (sitio sugerido por W3C)
www.w3schools.com/xml

[W3C02]
Página Web
«Web Services Definition Language 1.1»
www.w3.org/TR/wsdl

[W3C03]
Página Web
«Simple Object Access Protocol 1.1»
www.w3.org/TR/soap/

[WRO01.0]
Libro
«Professional C#». Simon Robinson et al.
2001. Wrox.
ISBN 186100499

Glosario

API
En inglés, Application Programming Interface significa interfaz de programación de aplicaciones. Permite a los usuarios utilizar primitivas de programación del sistema operativo.

APPLICATION DOMAIN
En la terminología de .NET, un dominio de aplicación es el medio por el cual el CLR permite a códigos distintos ejecutarse en el mismo espacio de un proceso.

ASSEMBLY
Es una unidad lógica que incluye el código intermedio de la aplicación e información adicional para la correcta gestión de la misma (metadatos).

B2B*i*
Acrónimo en inglés de *Business to Businees Interactions,* que significa interacciones de negocio a negocio.

CLR
Acrónimo en inglés de *Common Language Runtime*. Es el gestor en tiempo de ejecución de las aplicaciones .NET. Es un conjunto de código encargado de la carga de las aplicaciones, su ejecución y de proveer servicios de seguridad, administración de memoria, recursos, etc.

CLS
Acrónimo en inglés de *Common Language Specification* que significa especificación de lenguaje común. Define un conjunto mínimo de estándares que garantizan que el código generado es accesible por cualquier lenguaje que cumple CLS.

CTS
Acrónimo en inglés de *Common Type System*, que significa sistema común de tipos. Es una especificación del sistema de tipos de .NET a fin de permitir la interoperabilidad de los lenguajes. Contiene un conjunto de tipos básicos acordados y reglas de cómo definir tipos personalizados (como clases).

CODIGO ADMINISTRADO
Las aplicaciones que corren bajo el control de una máquina virtual (tal como el *CLR* de .NET) se dicen «administradas», de aquí surge el concepto de código administrado, en oposición a los conceptos de código no administrado y nativo.

CODIGO INTERMEDIO
Las aplicaciones que corren bajo el control de una máquina virtual (tal como el *CLR* de .NET) contienen código intermedio, el cual es una representación para dicha máquina abstracta. Se denomina intermedio porque generalmente se halla en medio del proceso de traducción de código fuente a código nativo.

CODIGO NATIVO
Se refiere al código de las aplicaciones en lenguaje de máquina, dependiente de la plataforma de hardware.

CODIGO NO ADMINISTRADO
Se refiere al código de las aplicaciones que no se ejecutan en el ámbito de una máquina virtual, accediendo directamente a los servicios del sistema operativo.

COLLECTIONS
Compact Framework ofrece un conjunto de clases en el *espacio de nombres* System.Collections que permite a los desarrolladores trabajar con colecciones o conjuntos de datos y manejarlos de manera simple de acuerdo a la naturaleza de su estructura de datos.

COMPACT FRAMEWORK
El *Compact Framework* de Microsoft es un marco de trabajo reducido diseñado especialmente para desarrollar aplicaciones para dispositivos móviles de capacidades limitadas de hardware y software. Será tratado con más detalles en el capítulo 2.

COMPILADOR JIT
JIT es el acrónimo para Just-In-Time o justo a tiempo. Indica compilación dinámica (en tiempo de ejecución) bajo demanda.

CONSTRUCTOR
Los constructores son métodos para la creación de objetos en memoria en estado de consistencia, es decir, no sólo asignan una porción de memoria al objeto sino que, además, el objeto es inicializado y dejado en condiciones apropiadas para su funcionamiento.

CPU
En inglés, Central Process Unit que quiere decir Unidad Central de Proceso.

DATASET
En la terminología de .NET, es una clase que representa la abstracción necesaria para manejar un contenedor de información offline (no necesita mantener la conexión con el origen de los datos).

DEBUGGER
Es el software que permite realizar el proceso de *debugging*, que consiste en la depuración de los errores que contiene una aplicación. Un entorno de desarrollo puede ofrecer facilidades como inspección de variables, punto de corte en la ejecución y ejecución paso a paso.

DESTRUCTOR
Los *destructores* son el concepto contrario de los *constructores*, su función es terminar el ciclo de vida de un objeto liberando todos los recursos utilizados: finalizando procesos abiertos, cerrando conexiones establecidas, devolviendo memoria solicitada, etc.

DISPOSITIVOS INTELIGENTES
En inglés, «Smart Devices». Pequeñas computadoras portátiles que proveen generalmente funciones de calendario, administración de contactos, toma de notas y pueden incluir otras aplicaciones como navegadores Web, reproductores multimedia, etc. En esta tesina asumimos que tienen almacenamiento permanente, capacidad de procesamiento y memoria de ejecución de programas.

DRIVER
Los drivers (o controladores) son piezas de software que permiten al sistema operativo la administración de un dispositivo hardware.

EAI
Acrónimo en inglés de *Enterprise Application Integration*, que significa integración de aplicaciones empresariales.

ESPACIO DE NOMBRES
El concepto de espacio de nombres (*namespace*) es una forma de agrupación lógica de clases de objetos, que permite ordenarlas de forma jerárquica en base a criterios establecidos, a fin de que el usuario pueda localizarlas y utilizarlas con facilidad.

EVENTO
Un evento es una acción o condición que se genera y determina que éste debe ser administrado por un manejador, el cual incluye el código necesario. El Compact Framework maneja un modelo similar al esquema productor-consumidor para el manejo de eventos en controles de interfaz de usuario.
EXCEPCION
Es la interrupción del flujo normal de ejecución de una aplicación, que se da de manera no deseada, y a veces, inesperada. No indica necesariamente un error de programación, puede deberse también a errores de ingreso de datos por parte del usuario o a condiciones fortuitas.
EXPRESIONES REGULARES
Las expresiones regulares son generadores de lenguajes regulares que pueden ser utilizados como reconocedores de los mismos. Permiten validar entradas de usuario por medio del reconocimiento de determinados patrones e informar cuando una cadena de texto no cumple la especificación del patrón.

FINALIZADORES
Son un mecanismo provisto por .NET para tratar de asegurar la liberación de recursos utilizados por un objeto fuera de lo que comprende la imagen de memoria (atributos y métodos) del objeto en sí.

GARBAGE COLLECTION
La «recolección de basura» es un mecanismo de administración de la memoria RAM que libera de forma automática y devuelve al sistema operativo el espacio en memoria solicitado previamente y que ya no es utilizado por una aplicación.
GUI
En inglés, Graphic User Interface que significa interfaz gráfica de usuario.

HEAP
El heap (o montón) es un área de memoria reservada en un proceso para la ubicación de memoria solicitada dinámicamente.
HTML
Acrónimo en inglés de Hypertext Markup Language, lenguaje por marcación de hipertexto, ampliamente utilizado y de uso estándar para la generación de páginas web.
HTTP
Acrónimo en inglés de Hypertext Transfer Protocol, que significa protocolo de transferencia de hipertexto. Es el protocolo cliente servidor de la suite TCP/IP utilizado en la World Wide Web para el intercambio de documentos *HTML*.

INTERFAZ
Las interfaces son una característica de los lenguajes orientados a objetos que permiten declarar que una clase implementa una funcionalidad determinada.
INTERNET INFORMATION SERVICES (IIS)
Es un software de administración de servicios Internet como servidores Web, FTP (protocolo de transferencia de archivos), administración de sitios, de aplicaciones Web, de seguridad, usuarios, etc.

LINKER
La función del linker (o enlazador) es crear una unidad de carga que consiste en programas y datos, para que el loader (cargador) del sistema operativo (o máquina virtual) los lleve a memoria de ejecución.
LOCALHOST
Localhost es un nombre que emplea IIS (*Internet Information Services*) para que las aplicaciones Web puedan ejecutarse localmente haciendo uso de un navegador, es decir, como si se tratara de una aplicación remota.

MANIFIESTO
Los metadatos se guardan en una sección especial denominada *manifiesto,* que contiene información acerca de los tipos de los miembros, y referencias a otras assemblies.
METADATOS
Información sobre los propios datos. En .NET, se refiere a los datos contenidos en una assembly.
MENU DE CONTEXTO
Son aquellos menús que aparecen en pantalla tras hacer clic con el botón derecho del mouse en Windows y ofrecen opciones adicionales de utilidad.

MIEMBROS
En la terminología de .NET, miembros son los métodos, variables y otros elementos que contienen las clases
MONOPROGRAMADO
En la terminología de los sistemas operativos, se refiere a aquellos que son capaces de ejecutar un único proceso completo por vez, sin posibilidad de cambios de contexto. Debe finalizarse la aplicación actual para dar paso a una nueva.
MULTITHREADED
El modelo de procesamiento *multithreaded* ofrece un pseudo paralelismo que es capaz de compartir el uso de un único procesador entre varios *threads* a intervalos de tiempo despreciables. Esto da al usuario la idea de procesamiento paralelo.
MSIL
En inglés, Microsoft Intermediate Language que significa lenguaje intermedio Microsoft.

NAMESPACE
Véase ESPACIO DE NOMBRES

PAL
En inglés, Platform Adaptation Layer que es una capa de abstracción entre el motor de ejecución CLR del .NET Framework y el sistema operativo.
PDA
En inglés, Personal Digital Assistant que significa Asistente Personal Digital.
P/INVOKE
P/Invoke (Platform Invoke) es una capa de interoperabilidad que posee el Compact Framework que permite al código administrado hacer invocaciones a librerías de enlace dinámico (DLLs). Esta funcionalidad se provee en el espacio de nombres *System.Runtime.InteropServices*
POCKET PC
Un dispositivo *Pocket PC* es un PDA basado en Windows CE que incluye aplicaciones personalizadas para cada manufacturera de hardware que las comercializa. Un sistema *operativo (Microsoft) Pocket PC* es aquél para un dispositivo homónimo.
PROXY (clase)
En la terminología .NET, una clase *proxy* contiene la lógica para traducir una invocación a través de Internet a un método que se halla en un servidor remoto.

QWERTY
Disposición de las teclas en teclados estándar de idioma inglés.

RAM
En inglés, Random Access Memory que significa Memoria de Acceso Aleatorio.
REFERENCIA
Las referencias son variables descriptoras de un objeto, situadas externas al heap, que poseen como atributos el nombre del objeto y su contenido es la dirección de memoria del objeto en el heap.
ROM
En inglés, Read Only Memory significa memoria de sólo lectura. El usuario o las aplicaciones no pueden escribir directamente sobre la memoria; sólo se permiten lecturas.
RPC
Acrónimo inglés de Remote Procedure Call, que significa invocación a procedimientos remotos. La idea central de RPC es permitir a programas situados en diferentes máquinas interactuar utilizando una semántica simple de *invocaciones* y *respuestas*, y así dar la idea de que ambas aplicaciones se encuentran en el mismo equipo.

SDK
En inglés, Software Development Kit significa kit de software para desarrolladores. Contiene herramientas de software, documentación y ejemplos para asistir al desarrollador en una tecnología determinada.
SERIALIZACION
La serialización es una función que ofrece .NET que permite «empaquetar» un objeto instanciado a un formato que puede ser binario para su envío en un flujo de datos determinado, como una conexión de red, un archivo, etc.

SIP
Es un mecanismo que utilizan los PDA (tipo PocketPC) para ingresar los datos, por medio de un teclado emulado por software.
SOAP
Acrónimo en inglés de Simple Object Access Protocol, que significa protocolo de acceso a objetos simples. Su función es proveer un mecanismo para intercambiar información estructurada y tipificada entre dos puntos de conexión, en un entorno distribuido utilizando *XML*.
SQL SERVER CE
Es una versión reducida de SQL Server de la compañía Microsoft, para el uso en dispositivos móviles de capacidades limitadas.
STACK
En inglés, stack significa «pila». Se refiere a una sección en memoria del sistema operativo, proceso o máquina virtual que se utiliza para la ejecución de unidades de programa. Se accede siempre a través del último elemento apilado.
STYLUS
El stylus es un puntero (dispositivo hardware) que permite seleccionar opciones en pantalla en un dispositivo Pocket PC. El usuario toma el stylus con la mano y hace clic sobre la pantalla para activar la opción que desea.

TIPO
Carlo Ghezzi [GUE98] define el concepto de *tipo* como la especificación del conjunto de valores que pueden ser asociados a un miembro, junto a las operaciones válidas usadas para crear, acceder y modificar sus valores.
THREAD
Thread (o hilo) es una unidad de ejecución independiente contenida en un proceso.

UNIDAD DE PROGRAMA
Se entiende por unidad de programa a métodos, módulos o bloques que tienen características propias en cuanto a su ámbito y visibilidad.

W3C
El World Wide Web Consortium (www.w3c.org) es una entidad creada en 1994 para tratar de aprovechar al máximo las características que ofrece la WWW. Su misión es: lograr el acceso universal de la Web, mantener un ambiente de desarrollo para que los usuarios de la Web hagan un uso óptimo de los recursos disponibles y guiar el desarrollo Web teniendo en cuenta aspectos legales, comerciales y sociales.
WEB SERVICE
Los servicios Web XML son unidades discretas de código que manejan un conjunto de tareas. Las aplicaciones que los utilizan exponen una serie de servicios a sus usuarios o sistemas, a través de protocolos Web abiertos. Representan una nueva «plataforma» en la que desarrolladores pueden construir aplicaciones distribuidas teniendo a la interoperabilidad como característica de máxima prioridad.
WiFi
WiFi hace referencia a un conjunto de estándares de la familia 802.11x para la comunicación inalámbrica entre computadoras y dispositivos.
WINDOWS CE
Es un sistema operativo de la compañía Microsoft pensado para dispositivos que llevan software empotrado en general, entre ellos dispositivos móviles.
WSDL
Acrónimo en inglés de Web Services Description Language, que significa lenguaje para la descripción de servicios Web. Es un lenguaje XML para la descripción de servicios de red definidos como un conjunto de endpoints (puntos que se comunican) y mensajes.

XML
XML es un metalenguaje que se utiliza como un medio independiente de plataforma de hardware y software para la representación y descripción de datos. Permite ocultar las diferencias entre plataformas ya que es un estándar que se basa en texto para la representación de los datos. Los documentos XML utilizan una combinación de *elementos* y *atributos* para representar la información de manera jerárquica.

Requerimientos de Sistema

- Tener algunas de las plataformas que se exponen en la siguiente sección
- Aproximadamente[129] 1.5 MB es memoria de almacenamiento permanente:
  - 1.55 MB (ROM) en Pocket PC 2000/2002
  - 1.35 MB (ROM) en Windows Mobile para Pocket PC 2003 o dispositivos Windows CE .NET

Plataformas en las que se ejecuta

- Pocket PC 2000
- Pocket PC 2002
- Pocket PC 2002 Phone Edition
- Pocket PC y Smartphone basados en Windows Mobile 2003
- Windows CE .NET 4.1 o superior

Plataformas en las que actualmente no se ejecuta

- Windows CE[130] 3 o anterior
- Handheld PC 2000 y anterior
- Smartphone 2002

Requerimientos de las aplicaciones[131]

- Los requerimientos de memoria RAM varían según la aplicación. Microsoft ha estimado como media estadística 0.5 MB aproximadamente.
- Los requerimientos de almacenamiento permanente RAM varían según la aplicación. Microsoft ha estimado como intervalo frecuente entre 5 – 100 KB aproximadamente.

129. Datos de referencia, según compilación con un procesador ARM en Pocket PC.

130. Puede sorprender al lector el hecho que Pocket PC 2000 y 2002 se basan en la implementación de Windows CE 3 y sin embargo soportan al Compact Framework. Los dos sistemas operativos mencionados son los únicos derivados de Windows CE 3 que son capaces de correr el Framework.

131. Según datos obtenidos de http://www.msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf/FAQ/default.aspx#1.1

Capítulo 1 Dispositivos Móviles

Capítulo 2 Ambiente de Desarrollo de Aplicaciones Móviles

Capítulo 3 Desarrollo de Aplicaciones en Dispositivos Móviles

Capítulo 4 Aplicaciones Móviles con XML Web Services

Capítulo 5 Desarrollo de una Aplicación Móvil con XML Web Services