UNIVERSIDAD DE BELGRANO

Las tesinas de Belgrano

Facultad de Tecnología Informática

Diseño de la arquitectura de un portal web escalable

N° 448

Andrés Irmer

Tutora: Viviana Esterkin

Departamento de Investigaciones
Diciembre 2010

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

Índice

ÍNDICE DE ILUSTRACIONES

I - INTRODUCCION

I.I - Conceptos básicos

Como principal introducción creo conveniente definir 2 de los conceptos que son parte del título de este trabajo. El primer concepto a definir es “Escalabilidad” y se la puede definir como la capacidad que tiene un sistema de expandirse de forma rápida y sencilla. En el caso de los Portales Web, la expansión hace referencia a la capacidad de presentar más cantidad de páginas en un mismo intervalo de tiempo.[1]

El otro concepto a definir es “Portal Web”. Se lo puede definir como el “Punto de entrada a un conjunto de recursos que una empresa desea poner a disposición de los usuarios del portal. En algunos portales de consumidores, el conjunto de recursos incluye toda la red, pero, para la mayoría de las empresas, el conjunto de recursos incluye información, aplicaciones y otros recursos que son específicos de la relación entre el usuario y la empresa.”[2]

I.II - Objetivos de un Portal dentro de una organización

Hoy en día muchas empresas tienen la necesidad de extender sus negocios y sus circuitos administrativos a Internet. Uno de los principales objetivos de los Portales es convertirse en un nuevo canal de ventas por medio de la implantación del Comercio Electrónico en las empresas que venden productos o prestan algún servicio. Como objetivo secundario, las empresas desean implementar un Portal Web para disminuir gastos administrativos publicando funcionalidades que le permitan a los usuarios finales auto gestionar trámites personales de manera tal de disminuir la intervención de Call Centers y de trámites presenciales. Por ejemplo, en la actualidad, hay una corriente muy fuerte en todas las empresas de servicios de telecomunicaciones de implementar Facturas Electrónicas. Esto les permite disminuir notoriamente los costos de impresión y al mismo tiempo mostrar un compromiso con el medio ambiente.

Actualmente, no basta con implementar un Portal con tecnologías antiguas generando páginas estáticas para implementar funcionalidad y presentar contenidos. Las empresas buscan productos de Software maduros, confiables y escalables que le otorguen una solución integral a su problemática de negocio. Esta solución integral deberá contender un esquema de alta disponibilidad ya que las funcionalidades que se instalarán en el Portal se convertirán muchas veces en el Core Bussiness de la organización, por lo cual deberán funcionar 7x24. Si esto no llegara a ser así se presentarían pérdidas económicas. La solución deberá proveer también de un esquema de seguridad robusto ya que el Portal estará exponiendo parte de los sistemas de la organización a Internet. Y por último la solución tiene que otorgar mecanismos para que el Portal crezca tanto en funcionalidades como en contenidos de forma rápida y confiable.

II - LINEAMIENTOS BÁSICOS

II.I Objetivos generales de esta tesina

El objetivo principal de esta tesina será analizar y realizar el diseño de una arquitectura de un Portal Web con tecnología Java que tenga como características principales escalabilidad, robustez y alta disponibilidad. Se presentarán varias evoluciones de la arquitectura y las modificaciones necesarias para escalar de una arquitectura a la siguiente hasta llegar a una arquitectura que posea las características buscadas. De esta manera y **como objetivo secundario se busca demostrar que una empresa puede implementar una solución de Portal con los mínimos requisitos posibles y, a medida que lo requiera, pueda evolucionar su solución hasta llegar a una arquitectura escalable, robusta y altamente disponible**.

Esto se debe a que en numerosos casos las organizaciones no poseen los recursos económicos para montar la mejor arquitectura posible desde el comienzo del proyecto. Por lo cual se presentarán diferentes

[1] Fuente: WebSphere Application Server V6 Scalability and Performance Handbook. Página 5.
[2] Fuente: Glosario de Sun Java Enterprise System

esquemas de arquitectura con sus pros y contras. La arquitectura no solamente será a nivel de grandes componentes sino que se planteará la arquitectura de software que deberán tener los componentes a desarrollar para implementar funcionalidades específicas que requiera la organización. Estos componentes de software se diseñarán teniendo en cuenta los últimos estándares y marcos de trabajo de Java.

II.II Contexto de la tesina

Esta tesina se desarrollará bajó el contexto de implantar un Portal Web para una empresa de Telecomunicaciones que ofrece determinados productos y servicios. En el caso práctico (que se desarrollará más adelante en esta tesina), la primera problemática que se planteó es la de implantar una arquitectura inicial de forma rápida para cumplir ciertos necesidades del negocio para disminuir costos en la gestión y en la emisión de facturas, dándole la posibilidad a sus usuarios de autogestionarse y de poder obtener su factura por medio de un archivo desde la Web. Este será el punto de partida para la primera arquitectura a desarrollar.

II.III - Justificación

Las empresas se encuentran en la búsqueda de soluciones informáticas integrales que le permitan instalar de forma rápida y segura sus negocios en la Web. Las empresas están dispuestas a realizar una importante inversión en esta tecnología siempre y cuando la misma le provea de seguridad y de mecanismos que le permitan extender sus contenidos y funcionalidades de forma rápida ya que las necesidades del negocio cambian día a día. También buscan que estas soluciones sean escalables en el tiempo a medida que sus negocios crecen y que al mismo tiempo sean confiables y se integren fácilmente con otros sistemas legados. Si bien implementar este tipo de soluciones pueden requerir de una inversión inicial importante, las empresas esperan que una vez instalada la arquitectura, generar nuevos contenidos y nuevas funcionalidades sea menos costoso y más rápido de manera tal puedan obtener un retorno de la inversión inicial.

II.IV - Premisas

Las arquitecturas a diseñar deberán ser multi-capa con componentes distribuidos.

Una arquitectura multicapa divide todo el sistema en distintas unidades funcionales: cliente, presentación, lógica de negocio, integración, seguridad, etc. Esto asegura una división clara de responsabilidades y hace que el sistema sea más robusto, mantenible y extensible.

Se partirá desde la arquitectura más simple a implementar y sobre la misma se presentarán evoluciones agregando componentes y redundancia en los mismos hasta alcanzar un esquema de Alta disponibilidad que le permita al Portal otorgar servicio a un alto número de transacciones simultáneas y tenga tolerancia a fallos. Por cada arquitectura se diseñará un esquema conceptual y uno real utilizando productos de IBM.

Por cada arquitectura presentada se desarrollará:
- Esquema conceptual
- Descripción de componentes
- Tecnologías utilizadas
- Limitaciones
- Ventajas con respecto a la evolución anterior.

II.V - Alcance

- Análisis y diseño de las arquitecturas.
- Relevamiento e investigación de los productos WebSphere Portal, Tivoli Access Manager, WebSphere Application Server, IBM HTTP Server.
- Se expondrá como implementar las arquitecturas propuestas con componentes de WebSphere y Tivoli.

- Diseño genérico de componentes de software para implementar funcionalidades específicas.
- Descripción y marco teórico de los framework Java utilizados.
- Exposición de una implementación real.
- Exposición de problemas encontrados a nivel de arquitectura en la implementación real productiva. Herramientas de diagnóstico y planes de acción para cada problema encontrado.

II.VI - Limitaciones

- No se analizarán otros productos de software para implementar Portales.
- No se proveerán instructivos de instalación y configuración de los productos de Software de IBM.
- No se analizarán temas de alta disponibilidad ni clusterización a nivel de base de datos.
- No se realizará un análisis de la capa de integración ni de sistemas legados.

III - MARCO TEÓRICO

III.I - Arquitectura multicapa

Las aplicaciones empresariales modernas poseen sus responsabilidades divididas en un número de capas. La arquitectura más común y básica es el modelo que posee 3 capas que consiste en:

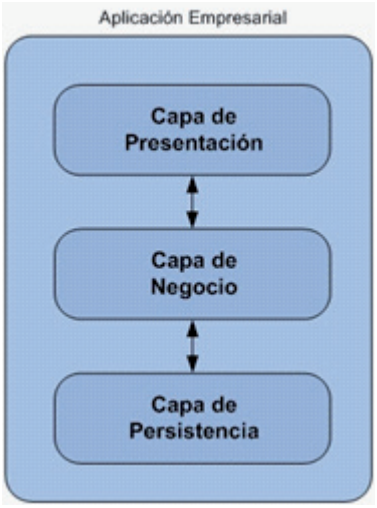


ILUSTRACIÓN I: APLICACIÓN EMPRESARIAL DE 3 CAPAS

- Capa de presentación: es la capa responsable de presentar al usuario final una interface y de interactuar con las acciones que realiza el usuario.
- Capa de negocio: es la capa encargada de ejecutar lógica de negocio. De acuerdo al “Glosario de Sun Java Enterprise System” la lógica de negocio o empresarial es el “código que implementa la funcionalidad básica de una aplicación en lugar de la integración de los datos o la lógica de presentación”
- Capa de persistencia (o de base de datos): es la capa encargada de persistir los datos de negocio interactuando generalmente con un motor de base de datos.

Dividir una aplicación en capas se utiliza en los sistemas de información para manejar más eficientemente la complejidad de los sistemas ya que cada una de la capas tiene un propósito particular.

La plataforma Java en su versión empresarial (Java EE, de ahora en más) provee servicios a una aplicación empresarial utilizando una arquitectura multi-capa.[3]

Los diseños de las arquitecturas de esta tesina se desarrollarán sobre un esquema multicapas de componentes distribuidos, siguiendo las mejores prácticas de J2EE y del desarrollo de aplicaciones para Portales. Las soluciones deberán contar una serie de capas lógicas, cada una con una función definida, para conformar la plataforma de ejecución del Portal, dicha plataforma se describe en el siguiente diagrama:

[3] Fuente: EJB Developer Guide. Páginas 7 - 9

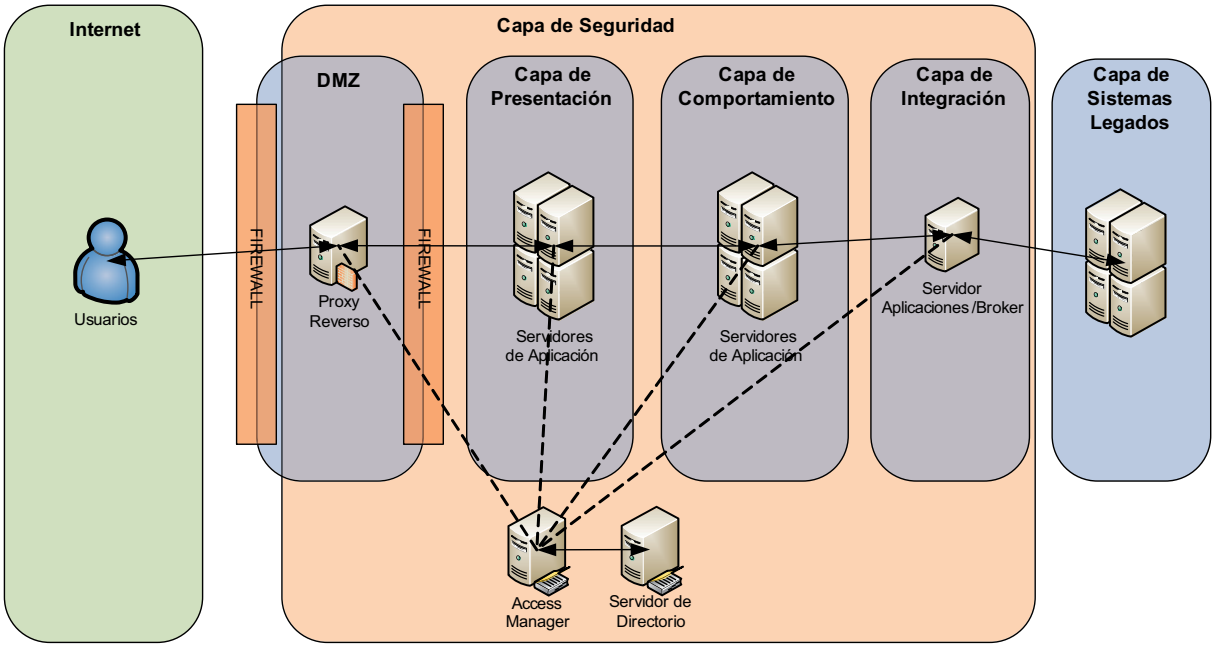


ILUSTRACIÓN II: CAPAS DE UNA APLICACIÓN DE PORTAL

La **Capa de Seguridad**[4] es el componente que resuelve la autenticación y autorización del Portal para los usuarios del mismo. La función de la Capa de Seguridad es fundamental desde el punto de vista de prevenir el acceso a contenidos privados o utilización de funcionalidad no autorizadas.

Esta capa estará implementada a grandes rasgos con componentes del producto IBM Tivoli Access Manager (o TAM). A lo largo de esta tesina se explicarán y se utilizarán los siguientes componentes:

- WebSEAL[5] (con la función de proxy reverso)
- Access Manager[6] (utilizando de éste los componentes Policy Server y Authorization Server)
- Directory Server[7] (un servidor de LDAP que sirve de repositorio de usuarios y credenciales para la solución), como elementos que conforman un contenedor de seguridad para el resto de los componentes del canal, desde el punto de vista que controla los accesos y se convierte en el repositorio único de datos de seguridad de la solución.

La **Capa de Presentación**[8] es la parte de la solución que despliega la visualización e implementa la navegabilidad del Portal. La función de la Capa de Presentación es la de composición de contenidos en función de los perfiles de los usuarios del Portal. Es fundamental su papel como filtro funcional y como utilidad de renderización de contenido consistente en cuanto a estilo, comportamiento y navegabilidad a través de todo el Portal y por lo tanto es vital el desarrollo de funcionalidades siguiendo estrictamente su arquitectura para asegurar su correcto acceso a la Capa de Comportamiento y evitar potenciales problemas de seguridad.

La **Capa de Presentación** estará implementada con el producto IBM WebSphere Portal Server, que es un paquete de software que incluye:

- IBM WebSphere Application Server: como servidor de aplicaciones J2EE
- IBM WebSphere Portal: la solución de renderizado de páginas de portal y contenedor de portlets, desplegada como una aplicación J2EE sobre el Application Server
- WebSphere Web Content Manager: otra aplicación J2EE sobre el Application Server para la administración de contenidos Web como parte de un circuito de aprobación y publicación entre editores.

[4] Develop and Deploy a Secure Portal Solution. Pag: 3 - 7
[5] Ver Anexo I
[6] Ver Anexo I
[7] Ver Anexo I
[8] Designing Enterprise Applications with the J2EETM Platform, Second Edition. Pag: 14 a 19.

La **Capa de Comportamiento**[9] es el componente de la aplicación que resuelve la lógica de negocio que necesita utilizar la capa de presentación, así como su parametrización y mapeos de información necesarios para interactuar con la Capa de Sistemas Transaccionales. Esta capa también se encarga de comunicarse con la base de datos y con el resto de los sistemas legados.

La **Capa de Comportamiento**[10] estará implementada por medio de un WebSphere Application Server como un repositorio de Enterprise Java Beans versión 3 (Este concepto se desarrollará más adelante en esta tesina)

La **Capa de Integración** es la plataforma tecnológica sobre la cual se implementan los servicios que permitirán la interacción de las aplicaciones del Portal Web con los sistemas transaccionales que tenga la organización. La función de la Capa de Integración es la de lograr una interfaz ordenada, de fácil mantenimiento y flexible. Brindando además la seguridad requerida para resguardar los datos en dichos sistemas. Además, la Capa de Integración aporta servicios de transformación de datos entre los diferentes puntos de acceso.

La **Capa de Integración** en un portal puede estar implementada con el producto IBM WebSphere Message Broker, que con sus componentes WebSphere MQ (para cola de mensajes asincrónicos) y WebSphere Message Broker Integrator (para transformación y enrutamiento de formatos de mensajes) permite interactuar a la capa de presentación, con la Capa de Orquestación y la Capa de Sistemas Transaccionales siguiendo el paradigma Enterprise Service Bus (ESB), siendo un punto fundamental para orientar la implantación de Portales hacia la filosofía Service Oriented Architecture (SOA).

La **Capa de Sistemas Legados** es el conjunto de sistemas actuales legados que soportan el negocio de la organización y a donde deben impactar las transacciones que genera la operatoria del Portal. Cabe aclarar que no son los sistemas actuales transaccionales como se encuentran hoy en día. Probablemente se requieran modificaciones en estos sistemas legados para implementar satisfactoriamente servicios que consumirán las aplicaciones del Portal.

III.II Escalabilidad

Conceptos básicos

Es muy importante para poder desarrollar esta tesina aclarar ciertos conceptos básicos que en definitiva son los objetivos que se quieren alcanzar en el diseño del Portal. La arquitectura final va a tener como requisitos las siguientes características:

ESCALABILIDAD[11]

Este concepto se refiere a la capacidad y configuración que tiene la arquitectura de un Portal Web de darle servicio a un número creciente de usuarios sin perder sus características de performance originales. Idealmente una arquitectura tendría que poder darle servicio a un número creciente de usuarios simplemente agregando más equipos o Servidores de Aplicación a la solución con el mínimo impacto posible en cuanto a las funcionalidades desarrolladas.

BALANCEO DE CARGA DE TRABAJO[12]

Esta característica se basa en que la configuración de la arquitectura se asegure que cada equipo o Servidor de Aplicaciones tengan una carga de trabajo justa. Es decir que compartan todo el trabajo de forma equitativa y que no haya un único servidor que tenga una carga muy alta mientras que el resto de los equipos se encuentre ocioso.

Si todos los equipos tienen una capacidad de procesamiento similar dado que su configuración de hardware es la misma, entonces todos deberían tener una porción equivalente de trabajo para ejecutar. Si

---

[9] Fuente: EJB Developer Guide. Páginas 8
[10] Fuente: WebSphere Application Server Version 6.1 Feature Pack for EJB 3.0
[11] WebSphere Scalability: WLM and Clustering - Página 3
[12] WebSphere Scalability: WLM and Clustering - Pagina 4 a 6

en cambio los equipos tienen capacidades diferentes entonces deberían ejecutar una porción de trabajo proporcional a sus recursos.

Si la cantidad de trabajo durante un periodo de tiempo cambia, entonces es esperable que el sistema se adapte naturalmente y por si mismo balanceando la carga de una forma equitativa o proporcional.

ALTA DISPONIBILIDAD[13]

El propósito principal de tener múltiples servidores en una arquitectura lleva al concepto de poder soportar alta disponibilidad del sistema. Este concepto se basa en que si alguno de los equipos de la arquitectura deja de ofrecer servicio por cualquier razón el sistema debe continuar dando servicio por medio de los servidores restantes.

La implementación de Alta Disponibilidad tendría que ser algo totalmente transparente al usuario que se encuentra utilizando el Portal Web. Es decir que si un servidor deja de funcionar, las solicitudes de un usuario tienen que ser re direccionadas a los servidores restantes sin tener ningún tipo de interrupción en el servicio ni requerir ningún tipo de acción.

Técnicas de escalamiento[14]

ESCALAMIENTO VERTICAL

Este tipo de escalamiento (también conocido como “Scaling Up”) consiste en añadir más recursos de hardware en el mismo equipo, en general, adicionando más procesadores y memoria al mismo. Si bien este tipo de escalamiento es fácil de aplicar puede llegar a ser un método costoso. De todas formas no se estaría evitando el problema de tener un único punto de falla. En caso de que el servidor quede fuera de servicio no importa cuántos recursos tenga se dejará de proveer servicio.

HORIZONTAL

También conocido como “Scaling Out”, significa agregar más nodos a la solución. Esto se puede lograr utilizando equipos de bajo costo. La principal ventaja es que por lo general es algo más económico y le permite a la solución tener capacidad de alta disponibilidad ya que si uno de los nodos de la solución se cae, el resto de la arquitectura se adaptaría para repartir la carga en los N-1 servidores restantes.

Criterios de selección de topologías[15]

Durante el desarrollo de la tesina se utilizan los siguientes criterios para planificar y justificar la aplicación de las arquitecturas propuestas:

SEGURIDAD

Los problemas de seguridad, por lo general, requieren la separación física del servidor HTTP (Web) de los procesos de aplicación de servidor, generalmente a través de la instalación de uno o varios Firewalls.

RENDIMIENTO

Implica reducir al mínimo los tiempos de respuesta para una operación determinada. En el caso de los Portales lo que se busca reducir es el tiempo en que la aplicación tarda en presentar una página web al usuario.

DISPONIBILIDAD

Este criterio requiere que la arquitectura tenga un cierto grado de redundancia con el fin de eliminar los puntos únicos de fallo. Si bien la escalabilidad vertical puede proporcionarlo mediante la creación de múltiples procesos, el equipo físico se convierte en un punto único de fallo. Por esta razón, una arquitectura que cumpla con alta disponibilidad necesita de una ampliación horizontal por medio de múltiples máquinas.

---

[13] WebSphere Application Server V6 Scalability and Performance Handbook - Página 21
[14] WebSphere Application Server V6 Scalability and Performance Handbook - Página 22
[15] WebSphere Scalability: WLM and Clustering - Capitulo 3

MANTENIBILIDAD

Si bien el concepto de mantenimiento se encuentra relacionado con la disponibilidad, hay una serie de cuestiones que deben tenerse en cuenta al implementar una topología que sea mantenible. De hecho, algunos de los conceptos de mantenimiento tienen objetivos que se oponen a la disponibilidad. Por ejemplo, un sistema sería fácil de mantener mientras que su número de instancias se mantenga al mínimo de manera tal que al momento de actualizar el software sea una tarea fácil y rápida. Si llevamos este ejemplo al extremo de lo ideal, desde el punto de vista de mantenimiento, habría que tener un único servidor de aplicaciones. Esto se contradice con el concepto de alta disponibilidad.

ESTADO DE SESIÓN

A no ser que la arquitectura tenga un único servidor de aplicaciones, compartir la sesión http y sus datos entre varios servidores que dan el mismo servicio, puede llegar a ser un factor determinante en la elección de una topología.

III.III Tecnologías a utilizar

Portales[16]

En la capa de presentación se utilizará la tecnología Portet. De acuerdo a la especificación de Portlet 1.0, un Portal se define como una aplicación basada en Web que provee de un conjunto de funciones y prestaciones. Las más comunes son:

- Contener la capa de presentación de un sistema de información
- Personalización. Es una característica que permite proveer de contenido especializado para diferentes tipos de usuarios. Es decir que el contenido generado en un Portlet tiene la capacidad de variar de un usuario a otro dependiendo de la configuración del mismo.
- “Single sign on”
- Agregación de contenidos de diferentes tipos. “Agregación” se lo puede definir como la acción de integrar contenidos de distintos tipos y orígenes dentro de una página Web.

Las páginas de un portal van a estar compuestas por un conjunto de Portlets creando contenidos y proveyendo funcionalidades a diferentes usuarios.

Un Portlet se lo define como un componente Web basado en tecnología Java que se encarga de procesar solicitudes de usuarios y de generar contenido de forma dinámica. Los Portlets son utilizados dentro de Portal como componentes encastrables dentro de una página proveyendo una capa de presentación a un sistema de información.

Un cliente Web va a interactuar con un Portlet mediante el paradigma Request/Response (Petición/Respuesta) que será implementado por el Portal. Este es el Paradigma clásico que se utiliza en la web. Tiene un esquema sincrónico en donde existe un cliente (en este caso un Explorador de internet o Browser) que solicita cierta información a un servidor o host de forma sincrónica. En otras palabras para cada solicitud (de ahora en más Request) existe una respuesta correspondiente de un servidor.

De acuerdo al Consorcio de la World Wide Web (W3C) un Request define una operación a ser llevada a cabo en una URL. Un objeto del tipo Request es la entrada principal de una aplicación para procesar una solicitud.

Un Portlet va a estar almacenado y su ciclo de vida será gestionado por un Contenedor de Portlets Se lo puede definir como un componente de un Servidor de Aplicaciones que tiene la función de ejecutar los Portlets que contenga proveyéndoles de un entorno de ejecución que necesitan. Un contenedor de Portlets recibirá los Request desde el Portal para ejecutarlos en un Portlet determinado que contenga.

Los Portlets tienen similitudes y diferencias con los Servlets. Estos constituyen otra tecnología Java muy utilizada en la capa de presentación de una aplicación.

[16] Java Portlet Specification Version 1.0. Capítulos PLT2 a PLT4

Las principales similitudes son:

- Ambos son componentes webs basados en tecnología Java
- Ambos son gestionados por medio de un contenedor
- Ambos generan contenido dinámico
- El ciclo de vida de ambos se encuentra administrado por un contenedor
- Ambos interactúan con un cliente web por medio del paradigma Request/response

Las principales diferencias son:

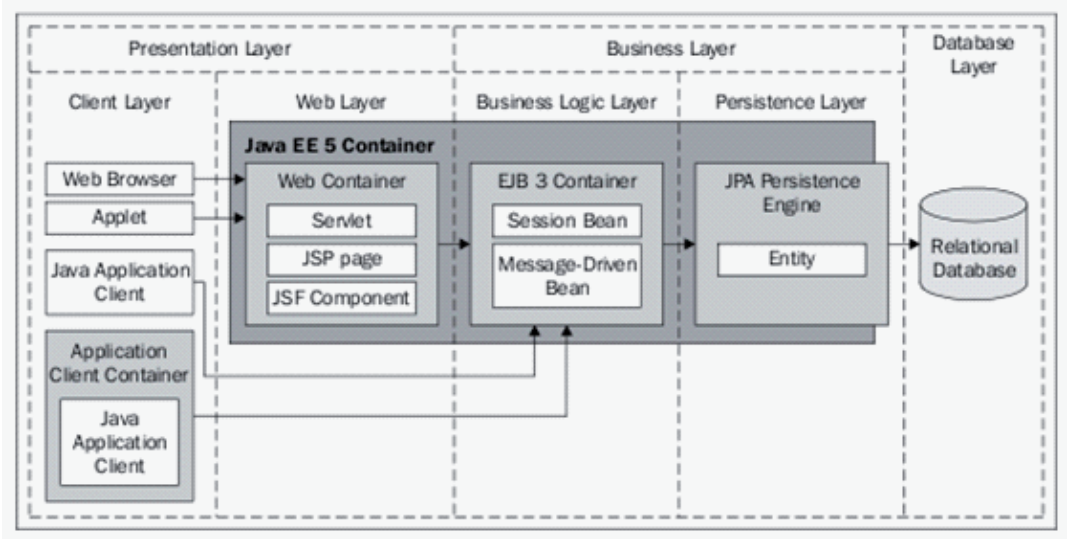
- Un Portlet no está asociado directamente a un URL.
- Los clientes Web interactúan con los Portlets por medio de un Portal.
- Los Portlets tienen un manejo de Request más especializado.
- Los Portlets tienen modos predefinidos y un estado que indica la función que el Portlet se encuentra ejecutando.
- Los Portlet pueden existir muchas veces dentro de una misma página.
- Un Servlet puede modificar la codificación de caracteres de la respuesta.
- Un Servlet puede modificar la configuración de encabezados HTTP en la respuesta.

En base a estas diferencias se decidió que los Portlets necesitaban ser un nuevo componente por lo cual un Portlet no es un Servlet. Sin embargo, para aprovechar la infraestructura existente, la especificación de Portlets se basa en funcionalidades provistas por la especificación de Servlets. Como por ejemplo el despliegue, el cargado de clases, manejo del ciclo de vida, manejo de sesiones y manejo de Requests.

EJB[17]

Enterprise Java Bean versión 3 (EJB3) es la tecnología que provee Java EE versión 5 para la capa de negocio. En esta versión de Java se subdivide la capa negocio en 2 capas. La capa de negocio propiamente dicha que contendrá todo el procesamiento de negocio y una segunda capa que contendrá todo lo concerniente a la capa persistencia.

En EJB3, los artefactos que se utilizan para el procesamiento de negocio se denominan Beans de sesión y Beans orientados a mensajes. Estos objetos son los que se deben desarrollar para una aplicación y son ejecutados en un contenedor de EJBs. En la capa de persistencia de EJB3 se utiliza un artefacto denominado “Entity”. Estos objetos se persisten en la base de datos utilizando un proveedor de persistencia o también llamado motor de persistencia. Este motor en realidad implementa una sub-especificación de EJB denominada Java Persistence API (JPA). La figura a continuación resume la arquitectura Java EE:



[18]

ILUSTRACIÓN III - ARQUITECTURA JAVA EE

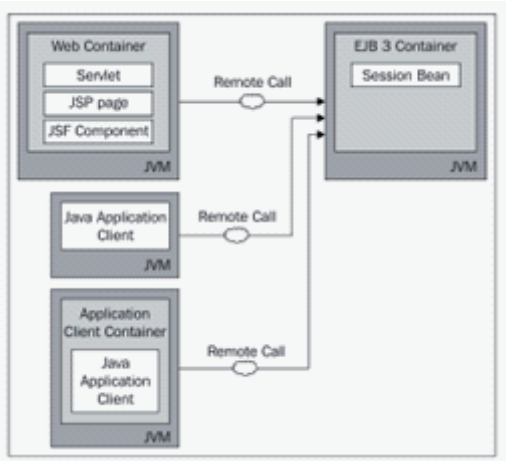
[17] EJB Developer Guide – Capítulo 1
[18] Gráfico extraído del libro EJB Developer Guide - Página 8

ARQUITECTURA EJB3

La arquitectura de EJB3 provee de un estándar para desarrollar aplicaciones de negocio distribuidas que sean orientadas a objetos y componentes. Los componentes que se desarrollan bajo este estándar se los conocen simplemente como EJBs. Estos componentes se sub-clasifican en objetos de sesión y objetos orientados a mensajes. Son denominados componentes en el sentido que al combinarse constituyen una aplicación de negocio de completa. Si estos componentes se desarrollan de forma correcta pueden ser reutilizados desde otras aplicaciones.

También se los clasifica como componentes distribuidos porque pueden residir en diferentes equipos y diferentes servidores de aplicaciones y pueden ser invocados de forma remota por un cliente que se encuentre en otro sistema.

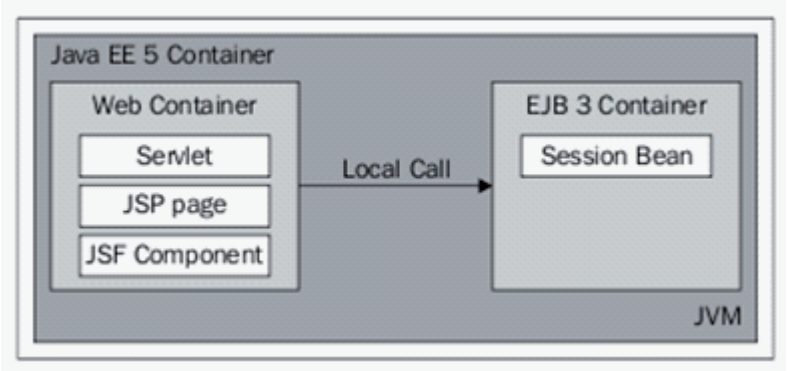
Un objeto de sesión debe tener una interface de negocio, esta interface puede ser remota o local. Un cliente remoto utilizará la interfaz remota para poder invocar a un método de negocio tal cual se muestra en el diagrama siguiente:



[19]

ILUSTRACIÓN IV: INVOCACIÓN DE UN EJB REMOTO

En cambio, si el cliente reside en la misma JVM en donde se encuentra el repositorio EJB, se debe invocar la interfaz local. El siguiente diagrama se muestra ese tipo de invocación:

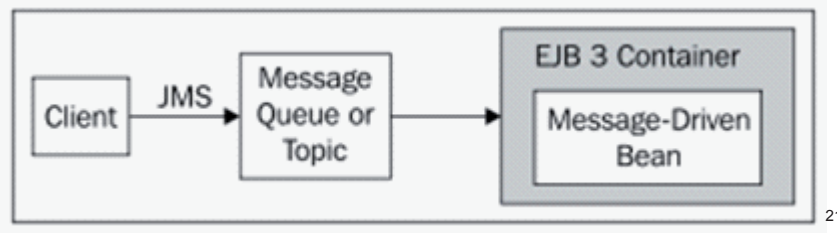


[20]

ILUSTRACIÓN V - INVOCACIÓN DE UN EJB REMOTO

Por último un objeto del tipo orientado a mensajes es aquel EJB que se encarga de escuchar una cola de mensajes y gestionar los mensajes que allí lleguen. Los mensajes pueden ser enviados desde un cliente remoto o local. El siguiente gráfico muestra como un cliente envía un mensaje a una cola JMS y el mismo es tomado por el Bean orientado a mensajes:

[19] Gráfico extraído del libro EJB Developer Guide - Página 9
[20] Gráfico extraído del libro EJB Developer Guide - Página 10



[21]

ILUSTRACIÓN VI - INVOCACIÓN DE UN MDB

Servicios provistos por un Contenedor EJB

Un contenedor EJB provee de un gran número de servicios. A continuación se listarán los servicios más importantes:

- Soporta procesamiento concurrente y todos los componentes de EJB3 son “Thread-safe” (Seguridad en hilos). Un objeto se considera Thread-Safe si puede ser invocado desde múltiples hilos de ejecución y no existe ningún tipo de interferencia entre dichos hilos.
- El contenedor provee de un servicio de nombres por medio del estándar Java Naming and Directory Interface (JNDI). Este mecanismo permite acceder a los objetos del contenedor por medio de un nombre.
- El contenedor EJB soporta la tecnología RMI-IIOP (Remote Method Invocation runo ver Internet Inter-Orb Protocol). Este mecanismo le permite a los clientes acceder a los e invocar a los objetos de sesión.
- El contenedor provee de un servicio para proveer transaccionalidad.
- El contenedor provee de una funcionalidad para planificar procesos.
- El contenedor provee de mecanismos para disponibilizar un objeto de sesión como un servicio Web.
- El contener provee de un motor de persistencia para interactuar con una base de datos.

MOTOR DE PERSISTENCIA JPA

El motor JPA tiene su propia especificación java y puede ser utilizado tanto dentro como por fuera de un contenedor EJB. Los principales servicios provistos por este estándar son:

- Manejo de entidades (Entity Manager): Provee servicios de persistencia, manejo de transacciones y manejo del ciclo de vida de las entidades.
- Mapeo de objetos relacionales (Object/Relational Mapping): Utiliza anotaciones de metadatos para establecer una relación entre un objeto y una tabla de base de datos relacional.
- Lenguaje de consulta propio denominado Java Persistence Query Language (JPQL): Es un lenguaje de consulta que permite recuperar información de una base de datos relacional.

IV - DESARROLLO

IV.I - Primer modelo de arquitectura de Portal

Objetivos principales

El principal objetivo de este primer modelo es poder montar una arquitectura que pueda ser utilizada para desarrollar y poner posteriormente en producción un prototipo del Portal. Este prototipo deberá soportar un número reducido de funcionalidades básicas y permitirá establecer lineamientos y estándares de desarrollo, estándares de consumo de recursos y métricas de acceso al Portal. En este prototipo inicial es recomendable que se desarrollen funcionalidades básicas de un Portal como Log-in, Registración y funcionalidades de consulta. No se recomienda implementar funcionalidades de comercio electrónico ya

[21] Gráfico extraído del libro EJB Developer Guide - Página 10

que la arquitectura, al ser un primer prototipo, puede ser inestable inicialmente. Esta primera evolución del Portal no soportará un número elevado de operaciones concurrentes.

Problemáticas inicial

Como la arquitectura a montar es un primer prototipo generalmente se asignan recursos de hardware limitados. En este momento tampoco se tiene una magnitud exacta del hardware que se necesita para dar servicio a un número X de usuarios. Por lo tanto la primera experiencia con la arquitectura se realizará con el mínimo Hardware requerido y en las siguientes iteraciones se analizará la necesidad de reforzar ciertos recursos o bien de distribuir la solución en varios servidores.

Esta primera iteración se basara la instalación de los componentes sobre una arquitectura de servidor PowerPC de 64 bits. Las especificaciones básicas de los servidores necesarios son las siguientes:

	Aplicativos	DMZ	Componentes de Seguridad
Denominación	SERVER A	SERVER B	SERVER C
Procesador	Power PC	Power PC	Power PC
Tipo de Procesador	64-bits	64-bits	64-bits
Clock Speed	1.8 Ghz	1.8 Ghz	1.8 Ghz
Memoria	12 GB	512 GB	512 MB
Procesadores Asignados	2	1	1
File System	4 GB	100 MB	300 MB

Modelo de la arquitectura propuesta

Conceptualmente la arquitectura de la solución deberá tener el siguiente diagrama que especifica los componentes de forma genérica pudiendo implementar con diferentes productos. En este caso el modelo de la solución se implementará por medio de la instalación y configuración de productos de la familia de IBM WebSphere y Tivoli para el esquema de seguridad.

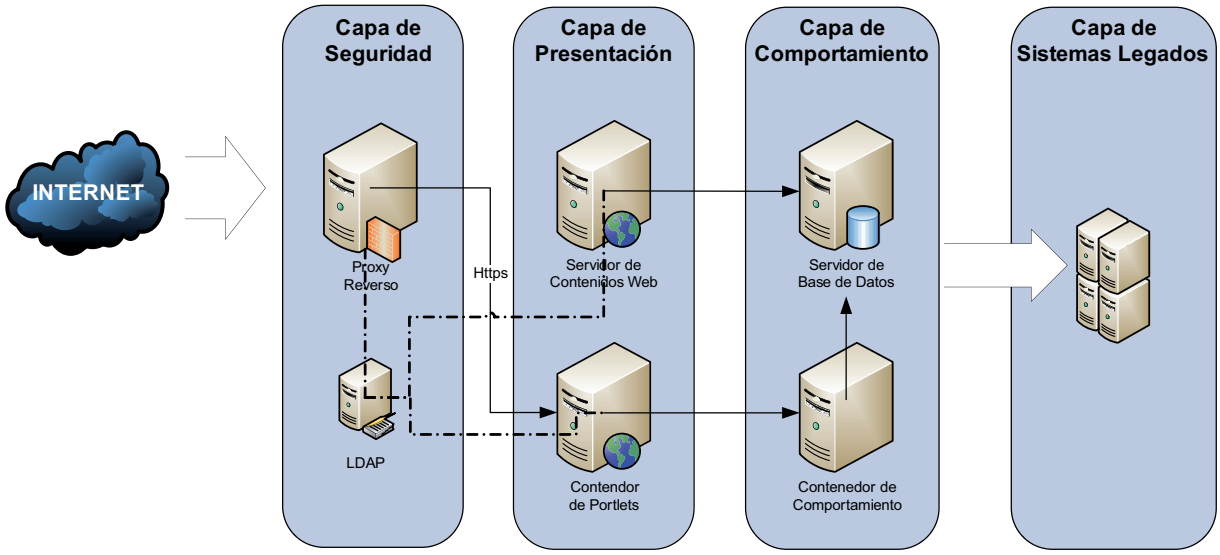


ILUSTRACIÓN VII - MODELO DE ARQUITECTURA I

Capa de seguridad

La arquitectura tendrá un único punto de acceso. Este punto se implementará por medio de un proxy reverso que tendrá como función recibir todas las solicitudes desde internet y direccionarlas al servidor de aplicaciones que corresponda. Se eligió utilizar un proxy reverso principalmente por las siguientes 2 funciones:

- Seguridad: El servidor proxy ofrece una capa adicional de seguridad ya que separa o enmascara el tipo de servidor que se encuentra por detrás del proxy.
- Encriptación (aceleración SSL): Cuando los sitios web son creados, generalmente la encriptación no la realiza el Web Server por una cuestión de performance. Se le da la responsabilidad al servidor proxy ya que por lo general este tipo de servidores vienen provistos con software para acelerar el cifrado SSL.

Otros motivos por los cuales es conveniente poner un proxy son las siguientes funciones aunque en este caso las mismas no se utilizarían:

- Balanceo de Carga: En el caso de esta arquitectura no hay múltiples servidores de aplicaciones que se encarguen de generar las páginas por lo cual esta función no se utilizará
- Caching: Un servidor de Cache puede liberar carga a un servidor web guardando en una memoria propia (llamada cache) todos los contenidos estáticos. Como por el momento no se tiene conocimiento de la carga que va a tener el servidor no se ve necesario la aplicación de esta función en un principio.

Capa de presentación

Esta capa es la encarga de componer y presentar páginas con contenidos y funcionalidades. Esta capa estará compuesta principalmente por 2 componentes.

El primer componente se denomina “Repositorio de Contenidos” y se lo puede definir como una aplicación genérica de almacenamiento que puede guardar texto e información binaria como imágenes y archivos. La forma de acceder a un repositorio de contenidos en Java se encuentra estandarizada por medio de la especificación JSR-170. Esta norma se define a sí misma como un estándar independiente de la implementación que busca proveer de un mecanismo para acceder a contenidos de forma bidireccional en un nivel granular dentro de un repositorio de contenidos. La especificación también define a un repositorio de contenidos como un sistema de administración de información de alto nivel que está compuesto por una colección de repositorios tradicionales de información. Este sistema implementa también servicios de contenidos como: versionado, motor de búsquedas, control de acceso de granularidad fina, categorización de contenidos y monitoreo de contenido.

El segundo componente a utilizar en esta capa es el Contenedor de Portlets. Este contenedor es el encargado de ejecutar Portlet y de proveerles el entorno adecuado de ejecución. Este componente contiene a los Portlets y administra su ciclo de vida.

Las razones por las cuales se eligió un Repositorio de Contenidos para esta arquitectura son:

- No hay que preocuparse por la forma en que se guarda la información en el repositorio (Base de datos, File System, archivos XML). Ya que la especificación provee de un mecanismo estándar para su acceso.
- El estándar provee de servicios para el almacenamiento y recuperación de los datos. La mayoría de los repositorios proveen de servicios avanzados como control de acceso unificado, motor de búsquedas, versionado, monitoreo, etc.
- Existen muchos productos maduros en el mercado que implementan Java Content Repository (JCR)

Las razones por las cuales se eligió utilizar un contenedor de Portlets para esta arquitectura son:

- El estándar permite desarrollar funcionalidades y utilizarlas como componentes a instalar en una página del Portal.
- Los Portlets se pueden integrar fácilmente con el Repositorio de Contenidos

- Existe una tecnología/técnica denominada “Bridging” que permite adaptar un servlet o una página web de manera tal que fuera ser invocada como un Portlet. Esto es beneficioso para que se pueden reutilizar otros componentes java ya desarrollados.
- Ciertas marcas proveen componentes y otros adaptadores para utilizar frameworks ya conocidos y maduros dentro de un Portlets. Tal es el caso de IBM que en su herramienta de desarrollo provee de Wizards que permite crear páginas de Struts o de Java Server Faces dentro de un Portlet.
- En el mercado se pueden encontrar Portlets ya desarrollados para integrar el Portal a otros sistemas de la organización.

Capa de comportamiento

Esta capa se encargará de proveer lógica de negocios a la capa de presentación. Es decir que cada vez que desde un Portlet se requiere obtener información del sistema (o del negocio), persistir información, ejecutar algún proceso de negocio (consulta, ventas, etc) se deberá invocar un proceso que se encuentre en este contenedor.

La tecnología elegida para implementar esta capa será un Contenedor de Enterprise Java Bean versión 3 (EJB3). Los motivos por los cuales se eligió esta tecnología fueron:

- La tecnología se encuentra preparada para arquitecturas distribuidas. Si bien en esta primera arquitectura se plantea instalar todos los contenedores dentro de un mismo servidor de aplicaciones. Es una buena práctica pensar en las posibilidades futuras de escalamiento de la aplicación en otro servidor y/o equipo.
- La arquitectura soporta el desarrollo, uso e instalación de WebServices. Característica que es muy importante al momento de integrar la lógica de negocios del Portal Web con los sistemas legados de la organización.
- La arquitectura de por si hace más fácil el desarrollo ya que los desarrolladores no tienen que entender o programar las transacciones a bajo nivel. Tampoco tienen que manejar hilos de ejecución, pool de conexiones u otras cuestiones de bajo nivel.
- La especificación 3.0 es mucho más sencilla que las implementaciones anteriores. Hoy en día existen entornos de desarrollo maduros que asisten a los desarrolladores en la creación de un EJB.
- La especificación contiene una sub-especificación denominada Java Persistence API. Esta especificación le provee al framework la capacidad de administrar datos de una base de datos (Persistencia de Datos). Se basa en un conjunto de técnicas que permite mapear una tabla con un Bean de Java. También provee mecanismos para realizar consultas a la base y cacheo de datos.

Capa de Sistemas legado

Esta capa está compuesta por el conjunto de sistemas actuales que soportan el negocio de la organización. Generalmente en estas aplicaciones es donde se deben impactar las transacciones que se generan en la operatoria del Portal Web. Estos sistemas, a su vez, deberán alimentar las funcionalidades del Portal que actúan como un Front-End para que los usuarios finales puedan realizar consultas.

Las tecnologías elegidas para interactuar con estos sistemas son las siguientes:

- Para realizar consultas sobre los sistemas legados se utilizarán Servicios Web WSDL cuya capa de transporte sea HTTP. Esta tecnología fue elegida ya que es un estándar maduro y provee muy buena interoperabilidad con otras tecnologías que no sean Java. Es muy probable que los sistemas legados de una organización no estén desarrollados en Java. Se eligió el protocolo HTTP como transporte ya que es la opción más común de implementación y las consultas a estos sistemas no requieren que la arquitectura persista las mismas. Si una consulta falla simplemente se despliega una pantalla al usuario final indicándole que vuelva a intentar más tarde
- Para realizar transacciones sobre los sistemas legados se utilizarán Servicios Web pero cuya capa de transporte sea Java Message Service (JMS). En este tipo de operatoria se utilizará la tecnología JMS para poder persistir los mensajes enviados a los sistemas legados de manera tal de no perder ninguna transacción en caso de que el sistema legado se encuentre fuera de servicio al momento de hacer una operación en el Portal. Por ejemplo si desde el Portal realizo se realiza una transacción de compra de un servicio. Es deseable que bajo ninguna circunstancia o contingencia, se pierda dicha invocación al sistema legado.

Implementación de la arquitectura con productos de IBM WebSphere y Tivoli

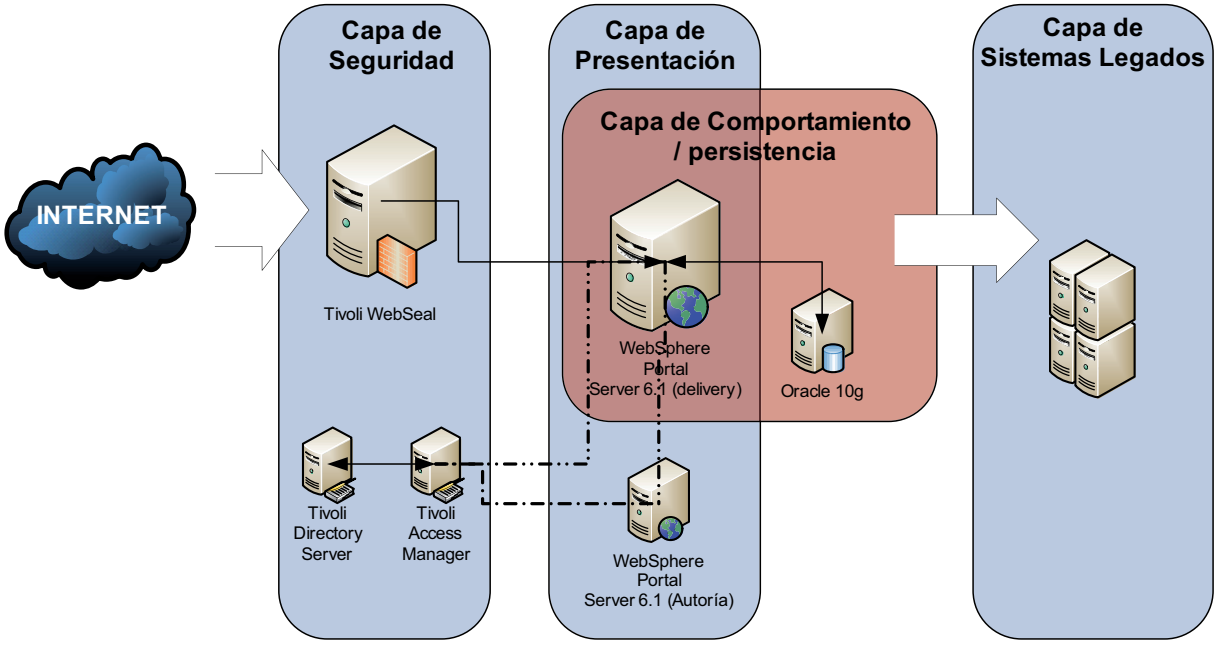


ILUSTRACIÓN VIII - IMPLEMENTACIÓN DE LA ARQUITECTURA I

Capa de seguridad[22]

La capa seguridad propuesta se implementará con productos de la familia de IBM Tivoli. Los componentes a utilizar son:

- Tivoli Directory Server (TDS): Este producto es un servidor de LDAP. Allí se almacenará el registro de los usuarios del Portal y los grupos/roles de la aplicación.

- Tivoli Access Manager (TAM): Este producto es un conjunto de componentes que dan servicios de seguridad. En esta arquitectura se utilizará para segurizar toda la solución. Desde el acceso al Portal Web hasta el acceso a los servidores de aplicación. Este producto se encargará de administrar la autenticación de usuarios, autorización de usuarios para ver determinado contenido y verificación de credenciales.

- WebSEAL: Es otro de los componentes de los componentes del TAM. Se configurará el mismo como un proxy reverso, es decir como punto de acceso al resto de la solución. WebSEAL también se encargará de interceptar todas las solicitudes que realiza un usuario. Si un usuario realiza una solicitud para visualizar una página, WebSEAL verificará por medio de TAM si dicho usuario tiene las credenciales adecuadas para visualizar dicha página. En el caso de que no las posea, el mismo será derivado a otra página.
  Muchas veces sucede que los usuarios de un portal agregan una determina página a los favoritos de su browser sin percatarse que para llegar a ese punto debe estar autenticado en el Portal. En otro momento, cuando utiliza la entrada de sus favoritos para volver a la página WebSEAL detecta que dicha página solamente puede ser visualizada por aquellos usuarios que se encuentran autenticados, por lo cual WebSEAL derivará al usuario a la página configurada para login y luego exitosamente se autentique será derivado a la página deseada.

Este software fue elegido por su fácil integración con los productos de WebSphere Portal que se utilizaran en la capa de integración. Esta herramienta además de ofrecer las funcionalidades básicas de un Proxy Reverso ofrece funcionalidades para realizar “Single-Sign-On”.

[22] Estos Productos se describen con más detalle en el Anexo I

Capa de Presentación

La capa presentación propuesta se implementará con productos de la familia de IBM WebSphere. Los componentes a utilizar son:

- WebSphere Portal: Es la aplicación encargada de la composición y presentación del Portal Web. El producto contiene principalmente dos aplicaciones que funcionan sobre un Servidor de Aplicaciones WebSphere.
  - La primera aplicación es el Portal Server propiamente dicho y es la que ofrece funcionalidades de contenedor de Portlets, herramientas de personalización y administración del Portal.
  - La segunda aplicación es el IBM Lotus Web Content Management que ofrece todas las funcionalidades para gestionar contenidos.

Para la capa de presentación se ha elegido la implementación de Portal de IBM WebSphere por las siguientes razones:

  - WebSphere Portal Server es una herramienta líder en la implementación de Portales.
  - Ofrece las funcionalidades más importantes descriptas en el modelo de arquitectura genérico. Ofrece las funcionalidades de Gestor de Contenidos y de Contenedor de Portlet.

Existen diferentes topologías en cuanto al modo de instalar y gestionar el producto. En cuanto al Portal en sí mismo se puede instalar como:

- Servidor único: Este tipo de topologías es la más sencilla de montar un Portal y consta, como su nombre lo indica, en un único servidor que da servicio.
- Cluster Horizontal: Este tipo de topologías es la más recomendada para entornos productivos ya que permite la instalación de múltiples nodos de la aplicación a lo largo de N servidores. Este concepto se explicará y se utilizará más adelante en esta tesina.

En esta arquitectura se eligió utilizar la topología de “Servidor Único” por la sencillez y rapidez de instalación. Aunque la principal razón fue que sólo se dispone de un servidor para instalar la solución.

También es necesario realizar una elección en cuanto a la topología que se utilizará para gestionar el Repositorio de Contenidos. La topología que se eligió fue la denominada “Two-Stage”. Está compuesta por:

- Servidor de delivery: esta instancia servirá para presentar el sitio Web al usuario final. Es decir el entorno productivo.
- Servidor de autoría: esta instancia se utilizará para crear o modificar contenido web que luego será migrado al servidor de delivery a través de un mecanismo de migración denominado “Sindicación”. Para lograr este objetivo, esta instancia tendrá instalado y configurado los portlet de Lotus Web-Content Manager. Dichos Portlet se encargan de administrar plantillas de autoría, componentes, workflows de aprobación y otros elementos necesarios para facilitar la creación de contenido web y gestionarlo ordenadamente.

WebSphere Portal puede trabajar en conjunto con diversos tipos de bases de datos. En esta oportunidad se ha definido utilizar Oracle por la confiabilidad de la misma.

A continuación se muestra un esquema más detallado que resume la composición de los capas de software de seguridad y de presentación y como se encuentran relacionadas entre sí:

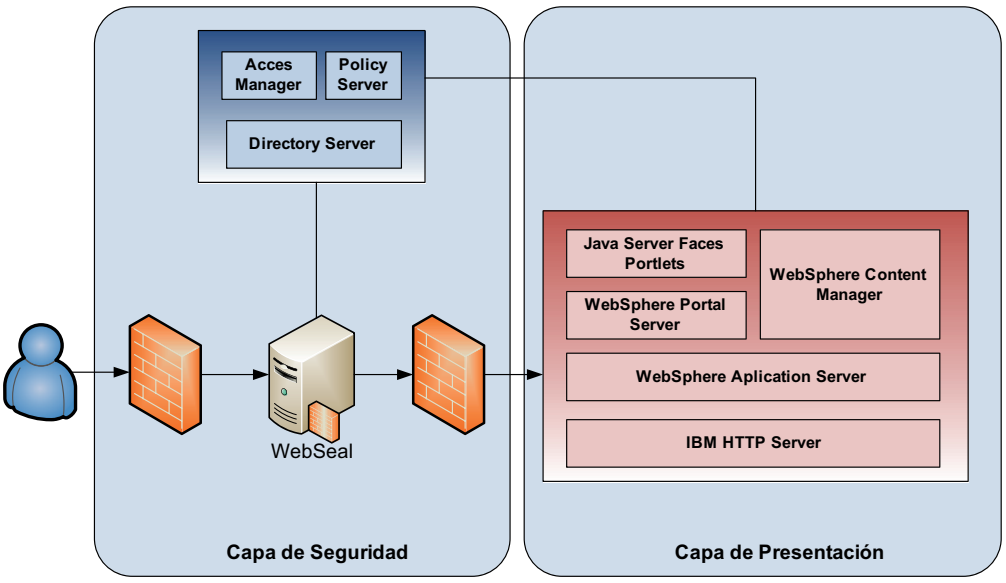


ILUSTRACIÓN IX - RELACIÓN DE LAS CAPAS DE SOFTWARE DE SEGURIDAD Y DE PRESENTACIÓN

Capa de Negocio

La capa negocio propuesta se implementará también con productos de la familia de IBM WebSphere. Para convertir un servidor de aplicaciones en un contenedor de Enterprise Java Bean 3 será necesario aplicar un “Feature Pack” sobre el servidor para darle la funcionalidad de contenedor.

En este Contenedor de EJB se instalarán los componentes desarrollados con lógica de negocios que será invocada desde la capa de presentación.

Las aplicaciones de este servidor utilizarán una base de datos Oracle, que servirá de base “transaccional”, donde se almacenará información de negocio, tal como datos de clientes, usuarios, etc.

Plan de Instalación

Es recomendable respetar el siguiente orden especificado a continuación. De lo contrario podría ser necesario rever uno o más pasos para ajustarlos a un nuevo escenario y realizar otras operaciones de configuraciones. Ninguna de esas actividades adicionales es contemplada en el listado.

LISTA DE SOFTWARE A INSTALAR/CONFIGURAR:

1. Instalación y configuración de Tivoli Directory Server (TDS).
2. Instalación y configuración de Tivoli Access Manager (TAM).
3. Migración de archivos de LDAP preexistente al TDS/TAM.
4. Instanciación y configuración de las bases de datos Oracle.
5. Instalación de todas las instancias de WebSphere Portal.
6. Configuración de cada instancia de WebSphere Portal.
7. Comunicación entre Portal y TAM.
8. Configuración de la instancia de WAS de capa de comportamiento.
9. Instalación y configuración de Workplace WebContent Manager.

Diagrama de servidores

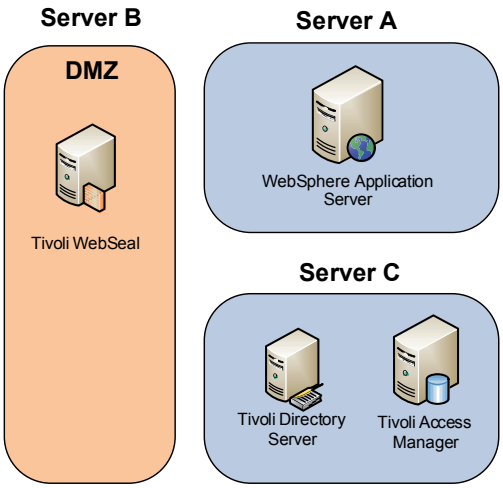


ILUSTRACIÓN X - DIAGRAMA SERVIDORES ARQ 1

Arquitectura de los componentes a desarrollar

Internamente un desarrollo pensado para la infraestructura anteriormente descripta deberá seguir determinados patrones de diseño para poder aprovechar la arquitectura. Al mismo tiempo la arquitectura de los componentes desarrollados deberá ser flexible para poder permitir un despliegue distribuido en caso de que se tengan varios servidores disponibles (repartidos en diversos servidores con propósito de seguridad o de alta disponibilidad).

Una aplicación desarrollada para desplegarse en el Portal Web debe contar con los componentes que muestra la siguiente figura para sacar provecho y poder desplegarse debidamente sobre la infraestructura del sistema.

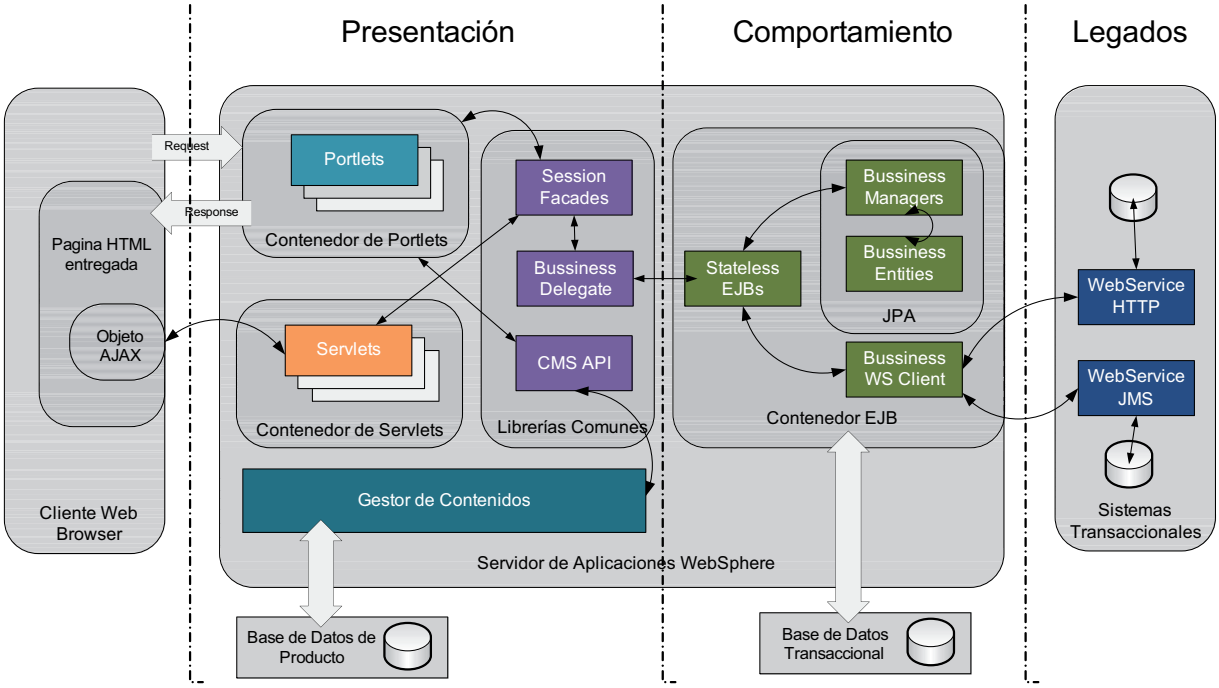


ILUSTRACIÓN XI - ESQUEMA DE COMPONENTES A DESARROLLAR

El Portal tendrá un conjunto de funcionalidades que estarán compuestas por uno o más Portlets que conforman su capa de presentación, dichos Portlets a su vez pueden utilizar y visualizar contenido que se encuentra en el repositorio de Contenido del servidor de Delivery (este, se actualiza, cuando se publica o

revisa un contenido con el Servidor de Authoring, como se explica anteriormente en la Capa de Presentación de la arquitectura). Cuando el Portlet debe acceder a la capa de negocio (o comportamiento) para realizar una operación dada, realiza una solicitud a la capa de negocio mediante el Business Delegate y un Session Facade específico. Este último es un patrón de diseño que encapsula la lógica y objetos de negocio exponiendo únicamente una interface. De esta forma se oculta la complejidad de la capa de servicios (En este caso la capa de comportamiento implementada con EJB3) y provee una forma unificada de acceder a la lógica de negocio en toda la capa de presentación. En cuanto al Business Delegate, en una arquitectura multi-capa y distribuida, se encarga de encapsular y ocultar la complejidad de comunicación entre diferentes componentes distribuidos. En este caso el Business Delegate se encargará de comunicarse con el contenedor de EJBs e instanciar el Stateless EJB indicado en la solicitud.

El Stateless EJB hará uso de los Managers y Entidades de Negocio para interactuar con el motor de base de datos. Tanto las entidades como los managers de negocio se implementarán por medio del Framework JPA (Java Persistence API). Este framework pertenece a una sub-especificación de EJB3.

Existirán al menos dos instancias de base de datos fundamentales para el funcionamiento del Portal: una la de Content & Configuration para el funcionamiento tanto del Content Manager como del Portal (Base de datos de producto) y la otra de datos propios de la solución del Portal a nivel funcional (Base de Datos Transaccional)

Además se utilizará el framework AJAX para la actualización dinámica de las páginas generadas sin necesidad de recargar las mismas. Para generar este componente AJAX se utilizará, el builder de Dojo Toolkit provisto por WebSphere. Para obtener información de los sistemas transaccionales el componente AJAX se comunica mediante un Servlet que sólo genera respuestas cortas (en comparación con las páginas del Portal) y que accede a la capa de comportamiento utilizando el mismo Business Delegate que los Portlets.

En el diagrama de secuencia a continuación se muestra la interacción entre componentes para una solicitud de Portlet.

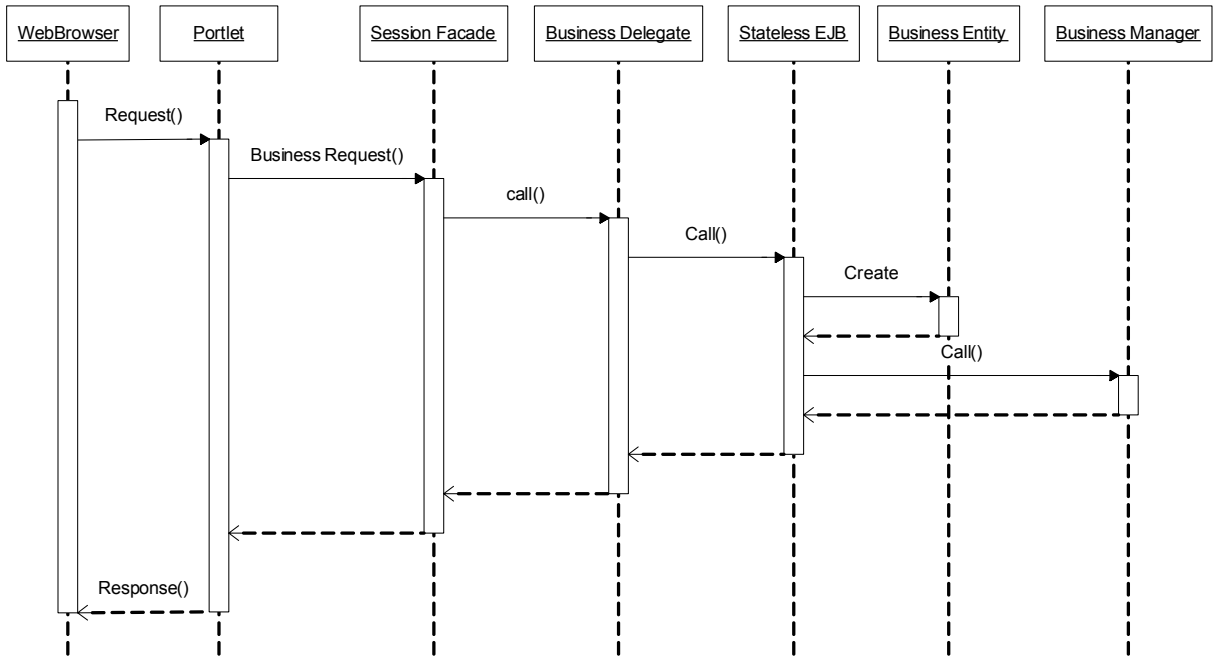


ILUSTRACIÓN XII - DIAGRAMA DE SECUENCIA

El objetivo de utilizar un diseño de aplicación tan altamente desacoplado como el descrito es el de poder distribuir los componentes según la necesidad de capacidad de procesamiento en cada capa.

El desarrollo de la funcionalidad del Portal se articulará en base diferentes Portlet, usando el criterio de división funcional (recomendado por mejores prácticas de portlet factory), utilizando fundamentalmen-

te dos tipos de Portlet (ambos compatibles con JSR-168) que se diferencian en función de la lógica de control de presentación, a saber:

- Generados con Portlet Factory (o Factorizados). Los Portlets factorizados utilizarán la lógica de control del runtime de Portlet Factory, que implementa un patrón Front Controller, para la composición de la presentación que implementan.
- Generados con JSF (o Compuestos). Los Portlets compuestos utilizarán la lógica de control implementada siguiendo los lineamientos de Java Server Faces, que permita implementar un patrón MVC, para la composición de la presentación que implementan.

El criterio para implementar una funcionalidad con un tipo de Portlet u otro será el de la complejidad y cantidad de controles estándares que requiera la pantalla.

Para un formulario sin mucha complejidad, y que se apegue a los formularios HTML estándares, los constructores de Portlet Factory permiten generar rápidamente la funcionalidad con un Portlet Factorizado.

Para un formulario que requiere un tratamiento especial de los datos o que utilice controles muy complejos para la presentación de la información, conviene implementar la funcionalidad con un Portlet Compuesto.

Las plantillas definidas para componer las páginas del Portal predeterminan el lugar donde se desplegarán las aplicaciones que implementan la funcionalidad del Portal. En cambio los Servlets se desplegarán como aplicaciones web separadas de la aplicación del portal, disponibles sólo para uso interno desde el propio portal.

Si la lógica de negocios requiere de interactuar con otros sistemas legados de la organización, dicha interacción puede encapsularse por medio de Servicios Web del tipo SOAP. Se utilizarán diferentes capas de transporte para los servicios de acuerdo a la operación que se requiera realizar sobre el sistema legado.

- Para realizar un impacto de datos en el sistema legado, se utilizarán los Servicio Web vía JMS. De esta forma se puede asegurar que el mensaje SOAP llegue a destino ya que la tecnología JMS puede persistir mensajes y asegura transaccionalidad.
- Para consultas de datos se utilizarán Servicios Web vía HTTP ya que si por algún motivo se pierde el mensaje SOAP no generará ningún tipo de inconsistencia. Simplemente se deberá volver a realizar la invocación.

IV.II Segundo modelo de arquitectura de Portal

Objetivos

El objetivo de este modelo será proveer de un diseño de una arquitectura que permita una escalabilidad vertical a medida que sea necesario. Además se buscará implementar mecanismos que permitan optimizar el uso de los recursos.

En un principio se estima que los usuarios que van a ingresar van a ser pocos pero lo deseable es que a medida que se agreguen funcionalidades al Portal, el número de usuarios que accedan al mismo aumente en alguna medida. A medida que las solicitudes de los usuarios aumenten comenzaran a escasear los recursos de la solución. Principalmente capacidad de procesamiento y memoria.

Modelo de la arquitectura propuesta

Conceptualmente la arquitectura de la solución deberá seguir el siguiente diagrama que especifica los componentes de forma genérica pudiendo implementar con diferentes productos. En este caso el modelo de la solución se implementará también por medio de la instalación y configuración de productos de la familia de IBM WebSphere y Tivoli para el esquema de seguridad.

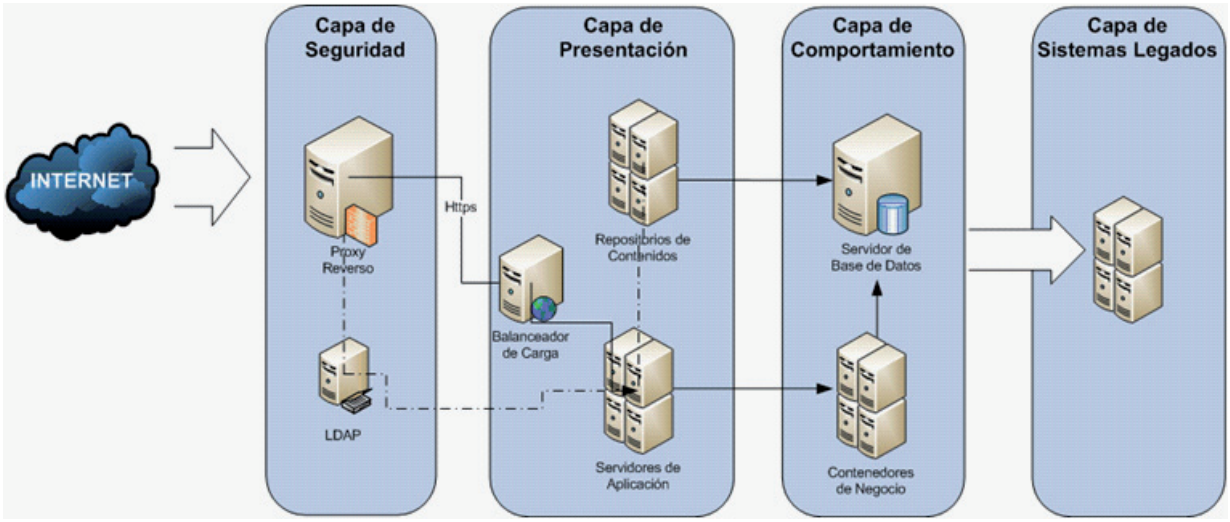


ILUSTRACIÓN XIII - MODELO DE LA ARQUITECTURA II

Capa de seguridad

Los componentes propuestos para esta capa serán exactamente los mismos utilizados en la capa anterior. Se decidió no realizar cambios en esta capa ya que este componente no requiere de muchos recursos para llevar a cabo sus funciones.

Capa de presentación

Para alcanzar el objetivo propuesto en esta capa, se deberán instalar múltiples instancias del Servidor de Aplicaciones de manera tal que se pueda distribuir la carga de procesamiento de forma equitativa. Más adelante se desarrollará la topología que deberán tener dichos servidores. Para que este esquema funcione se deberá agregar un componente denominado balanceador de carga que sea capaz de recibir todas las solicitudes de páginas Web y pueda enviar dicha solicitud al servidor de aplicaciones que en el momento tenga menos carga. La principal ventaja de este esquema es:

- En un futuro se puede agregar más servidores de aplicaciones de manera sencilla a la arquitectura. De esta forma la aplicación podría continuar escalando a medida que se necesite.

En esta capa también se deberá implementar algún componente que tenga la capacidad de guardar en un cache en memoria o disco todos los contenidos estáticos del Portal. Como contenido estático se entienden archivos de imágenes, hojas de estilo, archivos de código java script, etc. Los beneficios de instalar este componente son:

- Se quita carga del servidor de aplicaciones ya que si los contenidos estáticos son guardados en un cache y entregados al cliente por el mismo, el servidor de aplicaciones no tiene que utilizar sus recursos para recuperar el objeto desde el Repositorio de Contenidos. Existen soluciones en el mercado que permiten realizar cache a memoria y a disco. Por lo general se trata de un servidor HTTP que actúa en cierto modo como un proxy.
- Al tener ciertos contenidos en una capa más externa de la arquitectura, los tiempos de presentación de página se reducen.
- Los productos que existen hoy en día en el mercado permiten realizar estadísticas sobre el sitio web.

Capa de comportamiento

En esta capa se aplicará un concepto similar que en la capa anterior. Se deberán tener múltiples contenedores de lógica de negocio de manera tal que la carga entre los mismos pueda ser repartida.

Las ventajas de este concepto también son similares:

- En un futuro se puede agregar más servidores de aplicaciones con un contenedor de EJB, de manera sencilla a la arquitectura. De esta forma la aplicación podría continuar escalando a medida que se necesite.

Capa de sistemas legados

Cada uno de los sistemas legados posee su propia arquitectura que se encuentra fuera de esta tesina. Por lo cual se deberá analizar en cada uno de los casos como mejorar su rendimiento y disponibilidad.

Implementación de la arquitectura con productos de IBM WebSphere y Tivoli

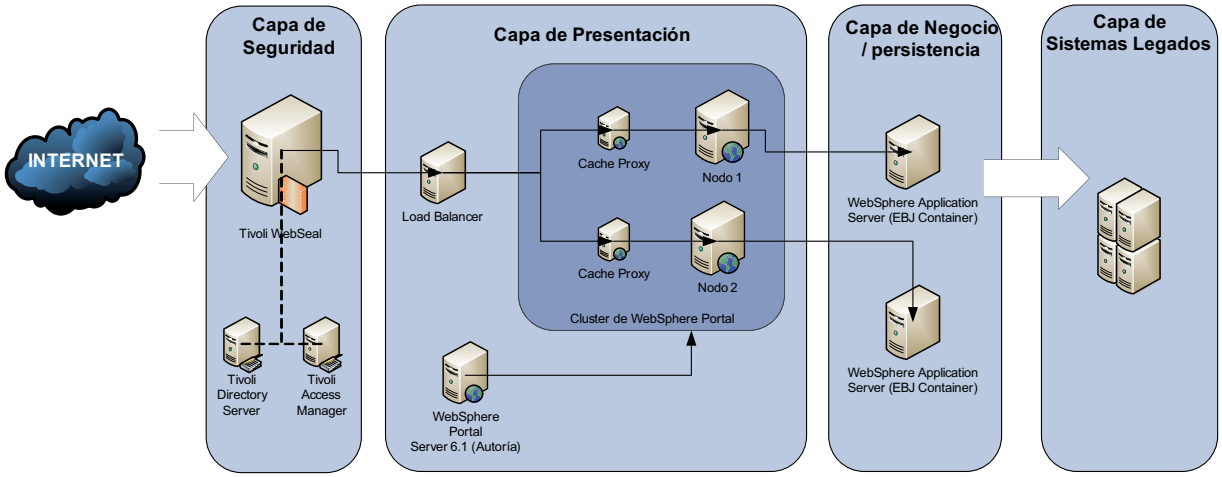


ILUSTRACIÓN XIV - IMPLEMENTACIÓN DE LA ARQUITECTURA II

Capa de seguridad

Esta capa se implementará con los mismos componentes de la capa anterior. Se deberá cambiar solamente una única configuración de WebSeal. En dicha configuración se deberán modificar las uniones entre este componente y el servidor de aplicaciones. Todas las uniones se deberán re-apuntar al componente balanceador de carga.

Capa de presentación

En esta arquitectura se eligió cambiar la topología en que se instala el WebSphere Portal Server. En lugar de utilizar un “Servidor único” se utilizó la topología “Cluster Horizontal” por las siguientes razones:

- Permite instalar múltiples nodos de la aplicación a lo largo de N servidores.
- Al permitir instalar múltiples nodos no es costoso crear un nuevo nodo si se llegara a necesitar por un crecimiento en la demanda.
- Permite balancear la carga entre múltiples servidores y aprovechar al máximo los recursos.
- En caso de que un nodo falle, el otro podrá satisfacer las demandas (aunque posiblemente con menor performance).

Para balancear la carga entre los diferentes nodos del Cluster se instalará el componente Load Balancer y Dispatcher. Ambos son parte del paquete WebSphere Edge Components.

Delante de cada servidor de aplicaciones se instala un componente adicional de la familia de WebSphere Application Server denominado “Cache Proxy” (Anexo I). Su función será la de guardar en un cache en memoria los contenidos estáticos. Este componente es parte de WebSphere Edge Components.

Arquitectura de los componentes a desarrollar

Internamente los componentes a desarrollar para una funcionalidad dentro del Portal deberán tener la siguiente arquitectura. La principal diferencia que se encuentra con respecto a la arquitectura anterior

es que el contenedor de EJB se encuentra separado del Servidor de Aplicaciones en donde se ejecuta la capa de presentación.

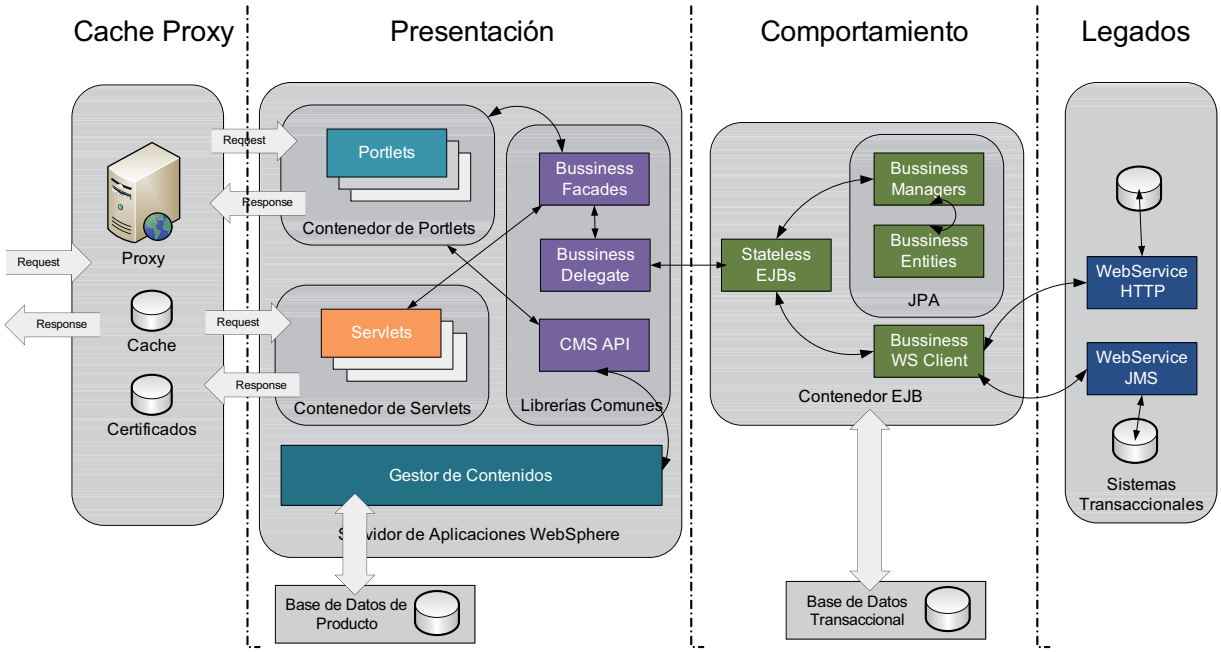


ILUSTRACIÓN XV - DIAGRAMA DE COMPONENTES PARA SEGUNDA ARQUITECTURA

Los componentes de la primera arquitectura fueron diseñados de manera tal de que puedan ser reutilizados en las siguientes iteraciones sin la necesidad de realizar modificaciones si deben ser instalados en un servidor diferente dentro de una arquitectura distribuida

El componente “Bussiness Delegate” será el encargado de 2 tareas. La primera será establecer una comunicación y una relación de confianza con el Servidor de Aplicaciones que posee el contenedor de EJBs. La segunda tarea constará en instanciar el EJB (Stateless) que requiere la capa de presentación para llevar adelante alguna tarea particular.

Diagrama de Servidores y Componentes de Arquitectura

Para esta solución se tienen solamente 2 equipos AIX que tienen las siguientes características:

	Aplicativo A	DMZ	Componentes de Seguridad	Aplicativo B
Denominación	SERVER A	SERVER B	SERVER C	SERVER D
Procesador	Power PC	Power PC	Power PC	Power PC
Tipo de Procesador	64-bits	64-bits	64-bits	64-bits
Clock Speed	1.8 Ghz	1.8 Ghz	1.8 Ghz	1.8 Ghz
Memoria	12 GB	512 GB	512 MB	24 GB
Procesadores Asignados	2	1	1	2

A diferencia de la arquitectura anterior, en este caso se dispondrán de 2 servidores en los cuales instalar el Cluster de Portal y los contenedores de EJBs. Básicamente se decidió instalar un nodo de Cluster junto con su EJB en cada servidor AIX para permitir una utilización máxima de los recursos (CPU

y memoria) de ambos servidores. El Proxy reverso se instalará en el equipo que se encuentra en la zona DMZ principalmente por cuestiones de seguridad, de manera tal de proveer una capa extra de seguridad sobre toda la solución propuesta.

A continuación se muestra en un diagrama que productos se instalarán en cada uno de los servidores.

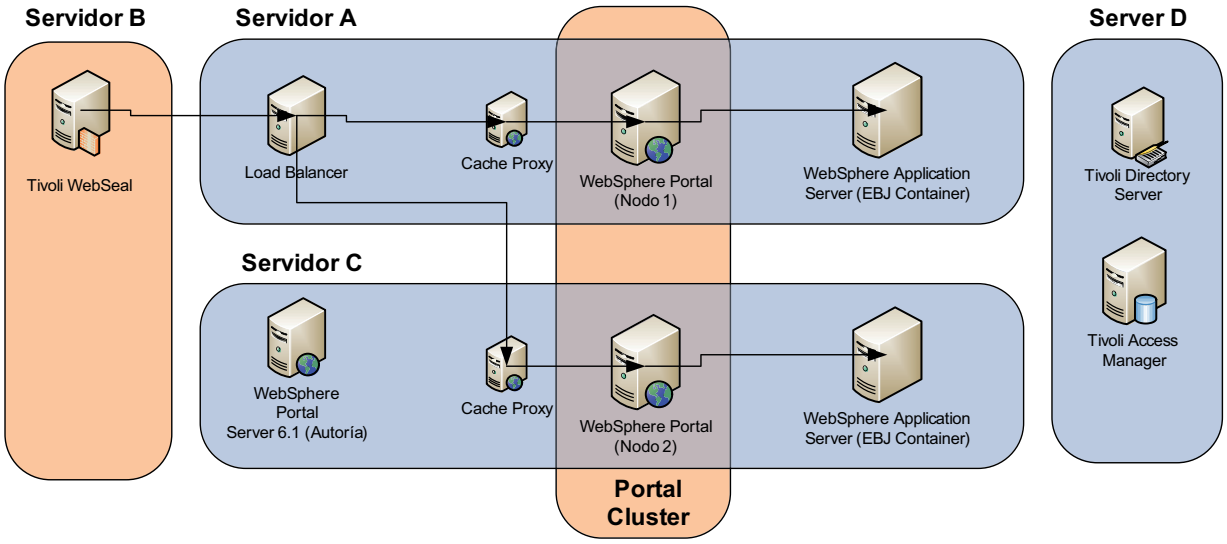


ILUSTRACIÓN XVI - DIAGRAMA DE SERVIDORES

Plan de instalación

A continuación se enumerarán las tareas necesarias para realizar la instalación de los componentes de la arquitectura:

1. Instalación y configuración de la base de dato Oracle que tendrá el repositorio de contenidos.
2. Instalar y Configurar WebSphere Network Deployment.
3. Instalar Nodo primario de WebSphere Portal
4. Configurar Portal Primario a la base de datos externa.
5. Federar nodo primario al Cluster.
6. Instalar Nodo secundario de WebSphere Portal.
7. Federar nodo secundario al Cluster.
8. Configurar Seguridad de nodos de WebSphere Portal.
9. Comunicación entre nodos de Portal y TAM.
10. Configuración de las instancias de WAS de capa de comportamiento.
11. Configuración de seguridad entre nodos Portal y WAS de capa de comportamiento.

Plan de migración

1. Instalar librerías comunes en el Cluster de Portal
2. Configurar Data Sources en los Servidores de Aplicaciones
3. Instalar los Portlets desarrollados de la solución en el Cluster de Portal
4. Instalar componentes EJBs en ambos servidores de aplicaciones
5. Exportar páginas de la arquitectura anterior e importarlas en el nuevo Cluster de Portal.
6. Sindicación de contenidos
7. Establecer una relación de confianza entre el Proxy Reverso (WebSEAL) con el Load Balancer
8. Modificar las uniones del Proxy reverso para que los Requests sean derivados al componente Load Balancer.

IV.III - Tercer modelo de arquitectura de Portal

En la arquitectura desarrollada en el punto anterior se detectan los siguientes problemas que se deberán mitigar en la última arquitectura:

- Existencia de puntos únicos de falla: La arquitectura se encuentra compuesta por varios componentes que ante una falla provocaría una indisponibilidad de toda la solución. Para mitigar este riesgo es necesario instalar y configurar todos los componentes bajo un esquema de alta disponibilidad de manera tal que el Portal continúe funcionando en caso de que alguno de los componentes deje de funcionar.
- Mayor complejidad en el diagnóstico de problemas: Cuando una arquitectura crece en la cantidad de capas que posee, al momento de diagnosticar cual es la causa cuando sucede un problema se hace cada vez más complejo. La forma de mitigar este riesgo es estableciendo alarmas y criterios que indiquen cuando la aplicación deja de estar disponible. Estas alarmas se deben configurar a lo largo de todas las capas para poder diagnosticar el problema más rápidamente.
- Alcance del Balanceador de carga: Este componente puede detectar cuando el Servidor de Aplicaciones del Portal no se encuentra respondiendo y puede derivar todas las solicitudes al otro nodo de la aplicación. Pero en caso de que exista un problema en el módulo de EJB que le presta servicio al nodo del Cluster, Balanceador no posee los mecanismos para detectar dicho problema.

Objetivos

El objetivo principal de este modelo será proveer de un diseño de una arquitectura que permita alta disponibilidad en todos los puntos de la solución. Los puntos únicos de falla detectados en la arquitectura anterior son:

1. Proxy Reverso (WebSEAL)
2. Load Balancer/Dispatcher
3. LDAP y Access Manager

Se denominan únicos puntos de falla dado que si por algún motivo dejaran de responder toda el Portal en su conjunto no funcionaría. Por lo cual se deberán instalar componentes redundantes de manera tal de que cuando alguno falle exista otro componente que se active para cumplir su función. Este cambio automático debe ser transparente e imperceptible para el usuario final.

Otro punto débil de la arquitectura es la relación que tiene cada uno de los nodos con su respectivo módulo de EJBs. En el caso de que algún servidor de aplicaciones de Portal deje de responder, el Load Balancer lo detectará y derivará automáticamente todo el flujo de procesamiento al nodo restante. En cambio, si existiera algún problema con un Contenedor de EJBs, el Load Balancer no percibiría dicho problema y nunca dejaría de enviar solicitudes a esa rama de la arquitectura. Esto tendría como efecto final errores que se serían visualizados por el usuario final. Por lo cual también se deberá mejorar la arquitectura de la capa de comportamiento de manera tal que siempre se encuentre disponible.

Modelo de la arquitectura propuesta

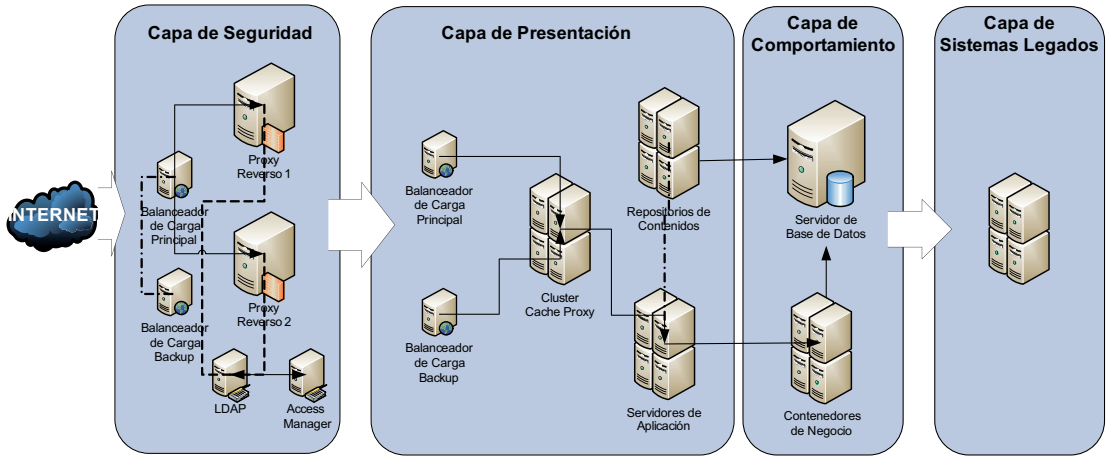


ILUSTRACIÓN XVII - MODELO DE LA ARQUITECTURA III

Capa de seguridad

Para alcanzar el objetivo propuesto, en esta iteración, se deberá implementar un Cluster de Proxies reversos. Para proveer de alta disponibilidad con sólo dos instancias es suficiente. Al tener instalado un Cluster de proxy reversos será necesario instalar delante del mismo un balanceador de carga de manera tal que reparta la carga entre ambos servidores. También será necesario instalar un balanceador de carga de backup que actúe en caso del que el principal falla. De no hacer esto se estaría introduciendo otro punto de falla en la arquitectura que iría en contra del objetivo planteado.

Capa de presentación

En esta capa se introducirán 2 cambios en la disposición de los componentes.

Se instalará un Balanceador de Carga adicional a modo de Backup delante del Cluster del Servidor de Aplicaciones. Este componente se activará en caso de que el principal deje de responder. Deberá existir algún tipo de mecanismo que les permita a ambos componentes tener conocimiento del correcto funcionamiento del otro.

En la arquitectura anterior cada nodo del Portal contaba con su propio Cache Proxy. En esta arquitectura se buscará mejorar las instancias Cache Proxy para que sean más eficientes.

Capa de comportamiento

Se cambiará la disposición de los Servidores de EJBs. En esta arquitectura se instalará un Cluster de Servidores de Aplicaciones que darán servicios a la capa de presentación. La diferencia con la arquitectura anterior es que cada nodo de Portal dejará de tener su instancia propia de EJB. En su lugar ambos nodos invocarán a los EJBs desde un único lugar.

Implementación de la arquitectura con productos de IBM WebSphere y Tivoli

A continuación se ilustra cómo se implementará la arquitectura final.

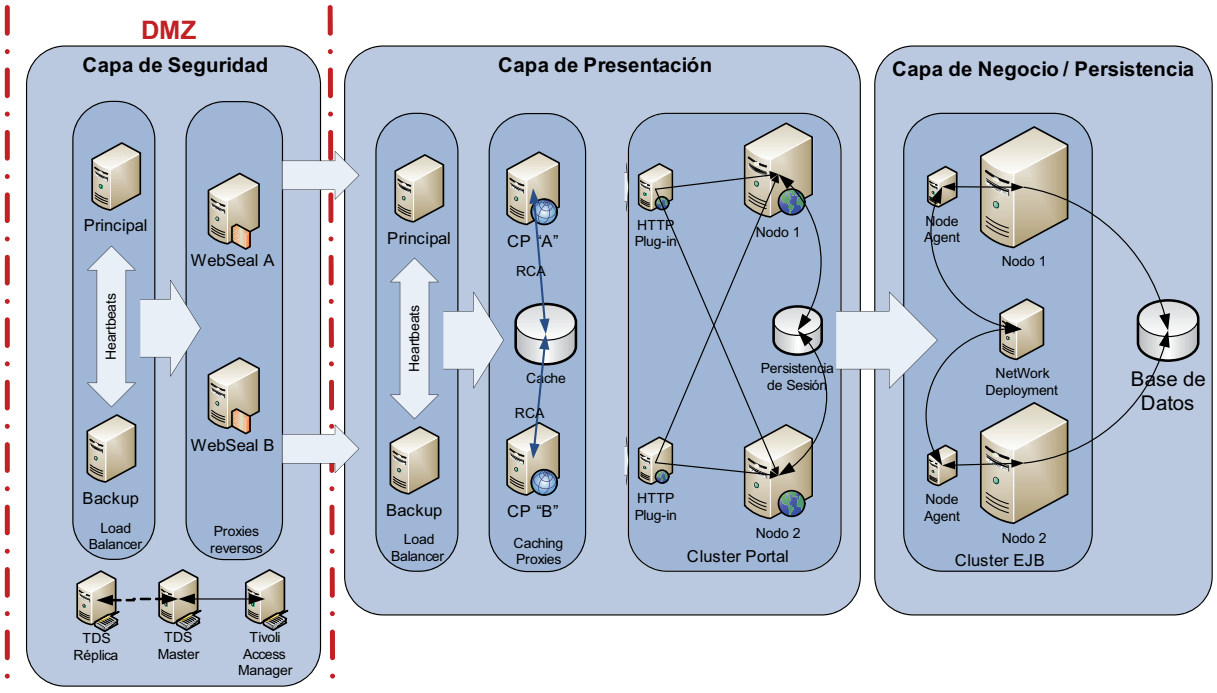


ILUSTRACIÓN XVIII - IMPLEMENTACIÓN DE LA ARQUITECTURA III

Capa de Seguridad

Para poder permitir alta disponibilidad en esta capa se implementarán 2 instancias del producto IBM Tivoli WebSeal. Ambas instancias tendrán una configuración idéntica.

Delante de las instancias de WebSeal se deberá instalar el componente denominado LoadBalancer de manera tal que balancee el trabajo entre ambas instancias. Para proveer un esquema de alta disponibilidad se instalará un LoadBalancer a modo de Backup (Más adelante se explicará la interacción entre ambos).

Al tener varios proxy reversos que atiendan los Requests de los usuarios se deberá tener cuidado con el manejo del estado de la sesión de cada uno de los usuarios conectados. Una sesión web se la puede definir como una serie de interacciones relacionadas entre un usuario particular y un servidor que tiene lugar en un periodo de tiempo. Cuando una sesión se encuentra establecida, el servidor de aplicaciones puede identificar el usuario asociado a cada Request y tiene la habilidad de recordar a un usuario específico entre numerosos Request.

Cuando no hay una sesión establecida, la comunicación entre el usuario y el servidor tiene que ser renegociada con cada Request que realice el usuario. Por lo cual mantener la información de una sesión mejora notablemente la performance de la aplicación en los siguientes casos:

- Para métodos de autenticación de usuarios como la autenticación básica (usuario y password), donde la información de autenticación es incluida en cada uno de los Request enviados a WebSeal, la información de sesión elimina la necesidad de validar el usuario y la password en cada uno de los Request.
- Para otros métodos de autenticación que requieren solicitarle al usuario que introduzca sus datos para autenticarse, el estado de la sesión elimina la necesidad de solicitarle al usuario sus datos de Log-in con cada una de los Request que realice.

Para manejar la sesión de un usuario a través de múltiples instancias de WebSEAL existen 4 posibilidades que se describen a continuación con sus pros y contras:

- No realizar acciones ante fallos: En esta posibilidad, si el balanceador de carga que se encuentra delante de WebSEAL logra mantener afinidad de sesiones, al ocurrir un fallo en el Proxy reverso simplemente no se realizan acciones. Esto traerá como consecuencia que el usuario deberá autenticarse de nuevo.
  Esta opción es la más fácil de configurar pero es la que tiene una peor experiencia de usuario en el caso de que WebSEAL falle o bien que el balanceador de carga no sea capaz de mantener la afinidad de sesiones.

- Inclusión de información de autenticación en cada Request: Existen métodos de autenticación que proveen cierta información en todos los Request que le permiten a WebSEAL realizar una autenticación automática con dichos datos. Si WebSEAL se configura para utilizar esta opción, cuando ocurra un fallo WebSeal tendrá la información suficiente en el Request para autenticar de forma automática al usuario.
  Este método es fácilmente configurable y tiene una buena experiencia de usuario. Sin embargo este método no permite el uso de ciertas funcionalidades de WebSEAL como la re autenticación y pkmslogout.

- Uso de Cookies “Failover”: este método es un mecanismo para re-autenticar un usuario y no un método para mantener la sesión. Estas cookies contienen información de autenticación encriptada que WebSEAL utiliza para validar la identidad de un usuario. Este tipo de cookie mantiene la siguiente información:

  - Información de credenciales de usuario
  - Tiempo de expiración de inactividad
  - Tiempo de expiración de vida

  La utilización de estas cookies presenta un riesgo de seguridad ya que si un atacante consigue capturar una cookie de un usuario puede utilizarla para autenticarse en el Portal hasta que pase

alguno de los tiempos de expiración. WebSEAL eliminará la sesión solamente cuando se transcurra el tiempo de inactividad o de vida configurados

Una ventaja de este método es que estas cookies pueden proveer de mecanismos para realizar Single-Sing-On.

- Servidor de Manejo de Sesiones (SMS): Este servidor se utiliza para que las instancias de WebSeal persistan la información de las sesiones en el Cluster de Portal. Cuando ocurre una falla, el nuevo WebSeal que tome el control va a poder obtener los datos de la sesión del usuario del SMS y por lo tanto no va a ser necesario solicitarle nuevamente los datos de autenticación al usuario. Las principales ventajas de utilizar un servidor para manejo de sesiones son:
  - Reduce el riesgo que se plantea en el uso de cookies del método anterior
  - El SMS provee información que el resto de los métodos no hace. Por ejemplo permite visualizar los usuarios autenticados en el Cluster en un momento dado.
  - Permite limitar sesiones concurrentes por el mismo usuario.
  - El SMS provee información histórica y de autoría.

Para este modelo de arquitectura el método elegido para mantener las sesiones de los usuarios será la implementación del Servidor de Manejo de Sesiones. Si bien su configuración conlleva más tiempo sus ventajas son muy superiores a los 3 métodos anteriores y es el único mecanismo que provee información de las sesiones de los usuarios.

Con respecto a la alta disponibilidad del LDAP, IBM Tivoli Directory Server soporta el concepto de servidor master y servidor réplica. Un servidor master contiene el directorio principal o Master desde el cual todo cambio que se realice es propagado hacia las réplicas existentes. Una réplica se la puede definir como un servidor adicional que contiene una base de datos idéntica a la principal o Master. Todo Servidor réplica solamente aceptará aquellos cambios que provengan desde el servidor master. Un servidor de réplica del proveerá a la solución un backup del LDAP que, en caso de que algo le suceda al LDAP principal, este podrá ejecutar cualquier búsquedas y proveer acceso a los datos[23]. Durante la ausencia del servidor principal no se podrán realizar operaciones de alta y eliminación del LDAP. Sin embargo WebSEAL va a poder utilizar alguna de las réplicas para autenticar a los usuarios.

En el caso de que el Servidor en donde se encuentra el LDAP principal puede promoverse a una de las copias como principal para reasumir el correcto funcionamiento de la solución.

Con respecto a la alta disponibilidad del Access Manager puede existir la posibilidad en donde el componente Policy Server falle y deje de responder. En caso de que esto ocurra el Portal no dejará de responder ya que en cada uno de los WebSEALs existe una réplica de la base de Datos de Autorización. De todas formas existe una forma de instalar un Access Manager secundario que se active al notar la no respuesta del primario. De todas formas, al no afectar el funcionamiento del Portal, el tema no se tratará en esta tesina.

Capa de presentación

El Balanceador de Carga de WebSphere posee una funcionalidad para proveer de alta disponibilidad. Básicamente permite configurar un Balanceador de Backup. Si el Balanceador de carga principal falla, el Backup tomará el control del balanceo de la carga de todo el Cluster.

Ambos componentes van a necesitar conectividad a los mismos clientes y al mismo Cluster de servidores de Portal. También necesitarán de conectividad entre ellos. Ambos componentes deberán ser instalados bajo sistemas operativos iguales y deben estar conectados a la misma red.

Un Balanceador de carga corre en un estado específico. El primer servidor corre en un estado de “Activo” y el Backup se encuentra en un estado de “Stand By”. Esto significa que el servidor “Activo” es el que se encontrará distribuyendo la carga. El Balanceador de Carga que se encuentra “Stand By” se encontrará

[23] Fuente: Axel Buecker, Anji Greene. Deployment Guide Series: IBM Tivoli Access Manager for e-business V6.0. (2008). IBM Red Books. Paginas (62-65)

monitoreando continuamente el activo. Si el servidor principal falla, el Backup realiza un operación de Failover cambiando su estado a Activo y comienza a balancear la carga. Lo que cambia es el estado del servidor pero no su rol. Por más que se tenga el estado activo continuará siendo el servidor de Backup.

Cuando el servidor principal vuelve a ser operacional pero en estado “Stand By” se puede configurar para que automáticamente tome el control y pase a estado Activo o bien que comience a monitorear el servidor de Backup a la espera de una falla.

Para monitorear el estado del servidor activo se utiliza un mecanismo denominado HeartBeats o latidos. En este mecanismo se envía una señal cada medio segundo al servidor activo. Si no hay respuesta en el transcurso de 2 segundos entonces un fallo es lanzado.

Con respecto a la utilización del Cache Proxy, es muy común que un Portal Web tenga más tráfico del que un único servidor pueda soportar. Por lo tanto en esta arquitectura se decidió simplemente agregar un servidor más de Cache Proxy. Si llegara a ser necesario se podrían agregar N servidores más. Sin embargo cuando existen múltiples servidores de Cache un contenido almacenado en un servidor puede superponerse con el mismo contenido pero almacenado en otro servidor. Esto trae como consecuencia redundancia innecesaria y evita ahorro en la utilización de recursos de red y de CPU ya que cuando un Cache requiere un contenido que no se encuentra en su propia memoria, por más que se encuentre en la memoria del otro servidor de Cache será necesario traerlo nuevamente utilizando recursos.

Para resolver este problema lo que se implementó en este modelo de arquitectura es utilizar un mecanismo que le permita a los servidores de Cache Proxy compartir sus contenidos en un único recurso de memoria. Este método se denomina Remote Cache Access (RSA). Básicamente es una característica del Cache Proxy que define un array de miembros que comparten un único repositorio. Dicho repositorio puede implementarse como un archivo o como un segmento de memoria.

Capa de Lógica de Negocio

Para alcanzar el objetivo de alta disponibilidad en esta capa se decidió implementar un Cluster de contenedores de EJBs. En la solución anterior cada Nodo de Portal tenía su propio contenedor de EJBs dedicado. El problema de la distribución de carga estaba solucionado en las capas superiores de la solución de manera tal que los EJBs iban a tener una carga proporcional a la que tenía el Nodo de Portal. Sin embargo cuando un módulo de EJB fallaba, el Portal al cual le presta servicio queda con un funcionamiento limitado ya que cada vez que se requiera ejecutar un método de negocio este fallara y el Balanceador de Carga que reparte el trabajo entre los nodos de Portal nunca podría determinar que existe un problema en las capas inferiores de la solución.

Para solucionar el problema de la alta disponibilidad, se instala en esta capa un Cluster de contenedores de EJBs. Los nodos del Portal no invocarían a un contenedor específico sino que invocarían a algún componente que sea capaz de derivar su solicitud a un EJB determinado, que se encuentre funcionando, por medio de alguna política específica.

Las características de Alta Disponibilidad y Distribución de Carga entre nodos de un Cluster de EJB se pueden alcanzar por medio de un Plug-in denominado “Workload Manager” (WLM) que se encuentra dentro del componente WebSphere ORB (Object Request Broker).

En el producto WebSphere Application Server Network Deployment, el balanceador de carga para EJBs se activa de forma automática cuando se crea un Cluster. No se requiere ningún tipo de configuración extra. El componente que tiene la responsabilidad de distribuir la carga entre los diferentes nodos el plug-in de WLM que encuentra en el ORB.

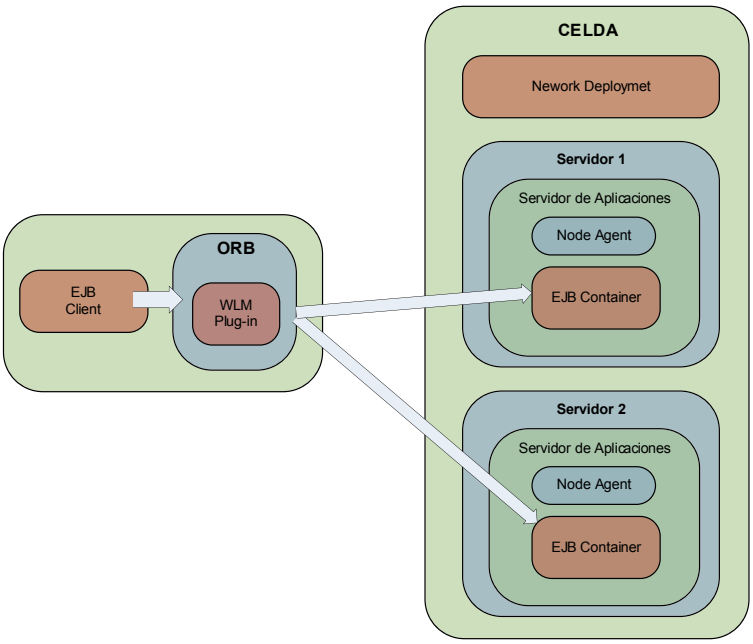


ILUSTRACIÓN XIX - DISTRIBUCION DE CARGA DE EJB ENTRE NODOS

Arquitectura de los componentes a desarrollar

Los componentes desarrollados para la arquitectura anterior se adecuan casi perfectamente a la nueva disposición de componentes. Únicamente se deberá realizar un mínimo cambio en el componente Business Delegate, este componente es el encargado de establecer una conexión con el contenedor de EJBs. Para poder utilizar las funcionalidades provistas por el producto para Balanceo de Carga y Tolerancia a Fallos se deberá especificar la dirección y puerto de todos los nodos del Cluster de EJBs. A continuación se muestra un ejemplo del código:

```
// Obtener el contexto remoto del servidor de EJBs

Hashtable env = new Hashtable();

env.put(Context.INITIAL_CONTEXT_FACTORY, "com.ibm.websphere.naming.WsnInitialContextFactory");
env.put(Context.PROVIDER_URL, "corbaloc::app1:9811,:app2:9812");

Context initialContext = new InitialContext(env);

// Instanciar la clase remota por medio de JNDI

try {

   java.lang.Object ejbHome = initialContext.lookup("cell/clusters/EJBcluster/BeenThere");

}

…
```

Diagrama de Servidores y Componentes de Arquitectura

Para esta solución se tienen solamente 3 equipos AIX que tienen las siguientes características:

	Aplicativo A	DMZ	Componentes de Seguridad	Aplicativo B	Aplicativo C
Denominación	SERVER A	SERVER B	SERVER E	SERVER C	SERVER D
Procesador	Power PC	Power PC	Power PC	Power PC	Power PC
Tipo de Procesador	64-bits	64-bits	64-bits	64-bits	64-bits
Clock Speed	1.8 Ghz	1.8 Ghz	1.8 Ghz	1.8 Ghz	1.8 Ghz
Memoria	12 GB	512 GB	512 MB	24 GB	24 GB
Procesadores Asignados	2	1	1	2	2

A diferencia de la arquitectura anterior, en este caso se dispone de 1 servidor más para instalar el Cluster de Portal y el Cluster de EJBs.

A continuación se muestra en un diagrama como se instalarán los componentes en cada uno de los servidores.

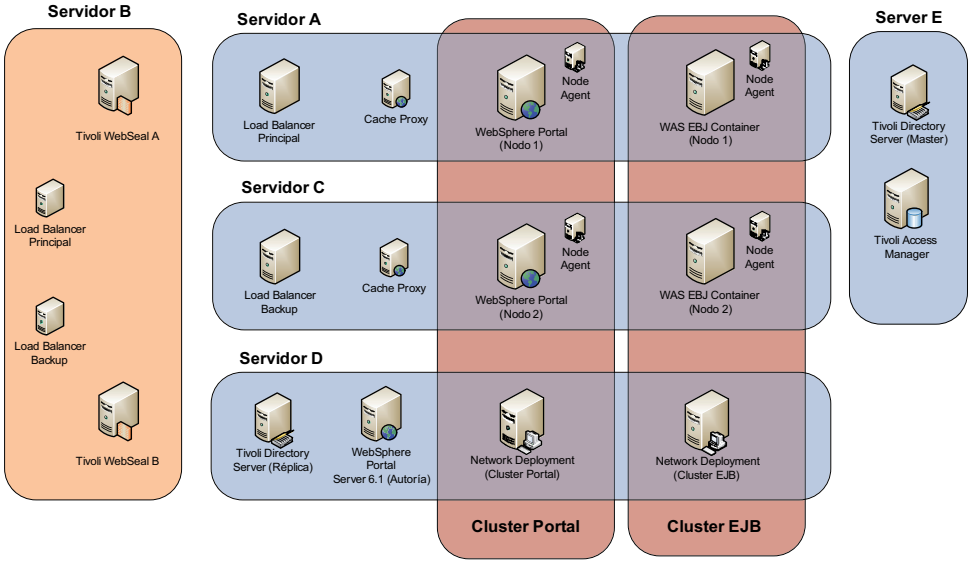


ILUSTRACIÓN XX - DIAGRAMA DE SERVIDORES DE 3 ARQUITECTURA

Plan de instalación

A continuación se enumerarán las tareas necesarias para realizar la instalación de los componentes de la arquitectura:

1. Instalar y Configurar WebSphere Network Deployment.
2. Instalar Nodo primario de WebSphere Application Server
3. Instalar Feature Pack para EJB 3
4. Configurar Nodo Primario a la base de datos transaccional.
5. Federar nodo primario al Cluster.
6. Instalar Nodo secundario de WebSphere Application Server
7. Instalar Feature Pack para EJB 3
8. Federar nodo secundario al Cluster.
9. Configurar Seguridad de nodos de WebSphere Application Server.

10. Clonar instancia de WebSeal en la capa de seguridad
11. Instalar LoadBalancer Principal y Backup en la capa de seguridad

Plan de migración

1. Por medio del Deployment manager instalar los EJBs desarrollados
2. Configurar “Remote Cache Access” de los Cache Proxies
3. Configurar Load Balancers en la capa de presentación
4. Configurar SMS en Cluster de Portal
5. Configurar Load Balancers en la capa de seguridad

IV.IV - Caso práctico

Descripción

En una empresa que presta servicios de telecomunicaciones se desarrolló un proyecto para crear un nuevo Portal Web que preste información y funcionalidades de autogestión a sus usuarios finales. Este proyecto surgió como una solución integradora de los múltiples sitios que tenía la empresa.

En base a los principales objetivos se implementaron las siguientes funcionalidades dentro del Portal Web:

- Auto-registración de usuarios
- Visualización de Facturas en formato PDF
- Adhesión a alarmar de vencimiento de factura
- Adhesión a servicio de factura sin papel
- Visualización de productos instalados
- Presentar información de productos y servicios
- Single Sing On contra otros sitios de la empresa que no fueron migrados al Portal

Como objetivos no funcionales se desea que el Portal Web cumpla los siguientes requisitos:

- Alta disponibilidad del Portal. Su funcionamiento debería ser 7x24
- Capacidad para que la solución escale en capacidad de procesamiento y memoria

Arquitectura inicial

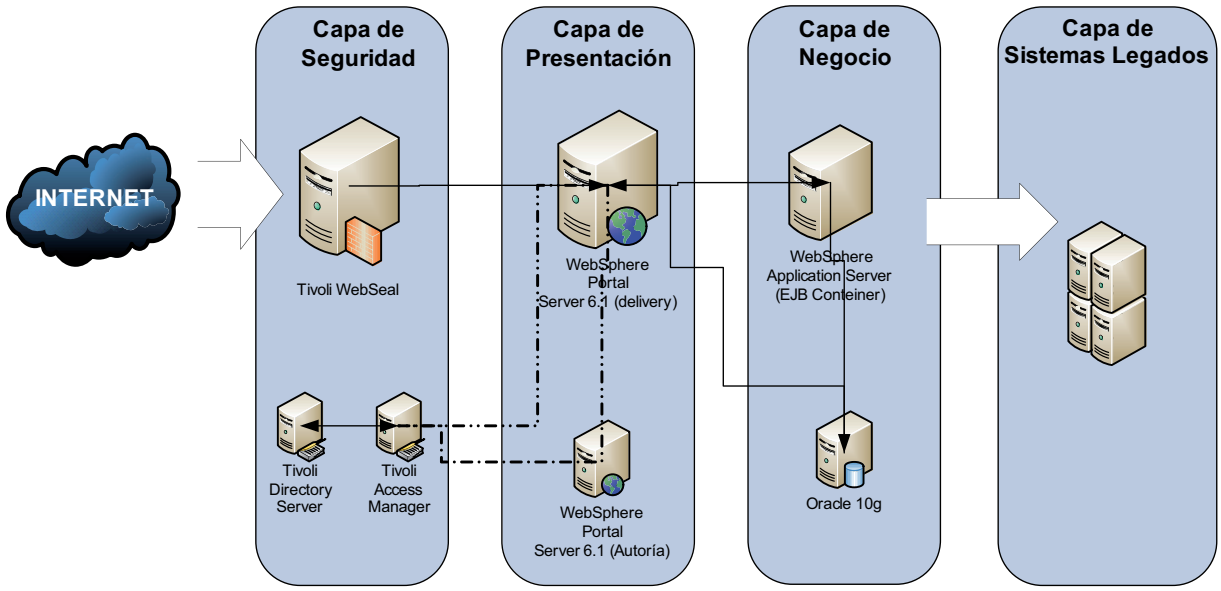


ILUSTRACIÓN XXI - MODELO DE LA ARQUITECTURA III DEL CASO PRÁCTICO

En este caso práctico se aplicó la arquitectura 1 con una modificación. La capa de presentación y la capa de negocio se encuentran en servidores de aplicaciones diferentes. A su vez los servidores de aplicación se encuentran en equipos diferentes. Se decidió realizar este cambio principalmente porque el cliente ya tenía instalado un servidor de EJBs. Tener separadas las capas de la solución permite aislar y diagnosticar más fácilmente los problemas cuando estos se presentan.

Capa se Seguridad

Esta capa se implementó con los siguientes productos de la familia de IBM Tivoli.

- Tivoli Directory Server (TDS): Este producto es el servidor de LDAP que funciona con una base de datos DB2. Este producto almacena los usuarios del Portal y los grupos/roles del Portal.

- Tivoli Access Manager (TAM): Este componente se encarga de administrar la autenticación de usuarios, autorización de usuarios para ver determinado contenido y verificación de credenciales.

- WebSeal: Este componente se configuró como un proxy reverso. Es el único punto de acceso al resto de la solución. Intercepta todas las solicitudes que realizan los usuarios. Si un usuario realiza una solicitud para visualizar una página, WebSEAL verificará por medio de TAM si dicho usuario tiene las credenciales adecuadas para visualizar dicha página. En el caso de que no las posea, el mismo será derivado a otra página.

Capa de Presentación:

Esta capa se implementó con WebSphere Portal 6.1 con la siguiente configuración:

- Servidor único: Es la topología más sencilla de montar un Portal y consta en un único servidor que da servicio.
- En cuanto a la topología para la gestión de contenidos se está utilizando la denominada “Two-Stage”. Que utiliza un Portal para Authoring y otro para Staging.

Capa de Negocio

Esta capa se implementó con un Servidor de Aplicaciones WebSphere con soporte EJB 3 para poder utilizar un repositorio de EJB. El servidor tendrá una topología de servidor único.

Justificación

Para este caso práctico se seleccionó la arquitectura 1 principalmente por las siguientes razones:

- Se necesitaba construir un primer prototipo para verificar que la solución era viable y cubría las necesidades del negocio. Si el prototipo cumplía satisfactoriamente con las pruebas de usuario y de estrés se convertiría en el primer entorno productivo.
- La construcción de esta solución debía realizarse en un tiempo acotado. No había suficiente tiempo ni recursos para realizar la investigación e instalación de una solución que cumpliera objetivos de carga y de alta disponibilidad a largo plazo.
- A corto plazo no se espera un gran número de usuarios concurrentes en el Portal. En caso de que el número de usuarios crezca y falten recursos se puede escalar la solución agregando más recursos a la misma (CPU, memoria y disco)
- Si bien la alta disponibilidad es deseable se acepta que no haya redundancia en los componentes mientras que no exista.

Estado actual de la solución

Este modelo de arquitectura hoy en día se encuentra productivo y funcionando. De todas formas la solución presenta los siguientes problemas y únicos puntos de falla.

- En la capa se Seguridad, al tener un único punto de acceso sobre el proxy reverso, se presenta un riesgo de indisponibilidad. Existieron incidencias en donde se registraron fallos de WebSEAL provocando una caída en toda la solución.
- En la capa de presentación también se presentan problemas de disponibilidad por la topología elegida. Al estar instalado como un servidor único se presentan los siguientes problemas:
  - Existen ocasiones en que el porcentaje de CPU consumido asciende de forma rápida y escalonada provocando una indisponibilidad de la solución. Este problema fue diagnosticado como un defecto del producto que está siendo investigado. Este problema podría ser mitigado por un esquema de alta disponibilidad.
  - El Portal tiene un límite para escalar a medida que aumenten la cantidad de usuarios. Por ser un servidor único no se podrán agregar más nodos a la solución a medida que sea necesario. Pero sí se podrá escalar momentáneamente la solución agregando más recursos siempre y cuando que la arquitectura de hardware lo permita.
- En la capa de lógica de negocio el principal problema que se encuentra es la falta de alta disponibilidad. Por el momento no se presentaron problemas de recursos de CPU pero si se presentaron numerosos de casos de indisponibilidad por problemas en el manejo de memoria. Si bien se agregó memoria al servidor esto no bastó para solucionar los inconvenientes.

Recomendaciones

En la actualidad se están realizando desarrollos que implementarán funcionalidades de comercio electrónico. Para cuando dichas funcionalidades pasen a ser productivo, el Portal deberá ser tolerante a fallas y deberá estar disponible todo el tiempo ya que la funcionalidad será un núcleo del negocio de la empresa. Por lo cual la arquitectura actual que posee el Portal no cumple con los requisitos futuros y se deberán tomar acciones para solucionarlo.

Lo recomendable para este caso práctico es migrar la solución de forma progresiva al esquema planteado en el modelo de arquitectura 3.

Para mitigar los problemas de la arquitectura actual se proponen los siguientes cambios que estarán orientadas a mitigar los problemas más importantes detectados. Estas recomendaciones acercan casi por completo a la solución a la arquitectura planteada en el modelo 3.

1. Instalar un Cluster de WebSphere Portal: Como primer objetivo se debería instalar un Cluster con 2 nodos para poder satisfacer la necesidad de alta disponibilidad y la capacidad de escalar la solución de forma sencilla cuando sea necesario.

2. Instalar un Cluster de WebSphere Application Servers para contenedores de EJBs. Si bien no hay necesidad de mayor procesamiento esta arquitectura mitigaría en gran medida los problemas de disponibilidad de esta capa de la solución. De todas formas se recomienda que se investigue y diagnostique el problema de fuga de memoria que presenta la solución.

3. Crear una instancia adicional de WebSeal e instalar un Balanceador de carga delante para aumentar la disponibilidad del proxy reverso. Clonar una nueva instancia y configurar un Load Balancer es un procedimiento rápido.

4. Para distribuir la carga entre los nodos del Portal se puede instalar de forma rápida otro balanceador de carga.

5. En un futuro se recomienda instalar Load Balancers de Backup para lograr mayor disponibilidad.

6. También se recomienda planificar la instalación de un Cluster de Cache Proxies tal cual se desarrolló en la arquitectura 3 para descargar carga en los servidores de aplicaciones.

A continuación se presenta un diagrama con las recomendaciones:

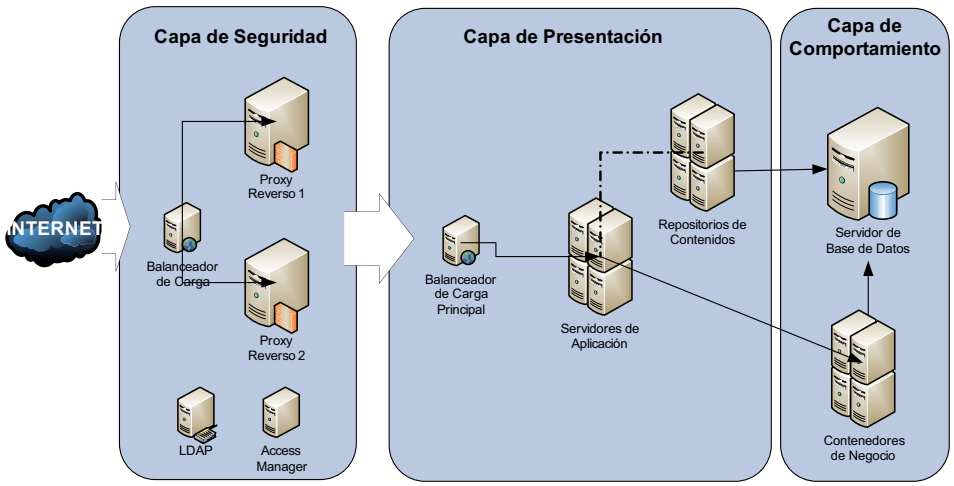


ILUSTRACIÓN XXII - IMPLEMENTACIÓN DE LA ARQUITECTURA DEL CASO PRÁCTICO

Ejemplos de componentes a desarrollar para implementar una funcionalidad

Se adjunta el anexo IV con un diseño técnico de ejemplo para una de las funcionalidades de este Portal Web. Dicho diseño sigue la arquitectura de componentes propuesta en esta tesina.

Problemas de arquitectura más comunes

Para analizar y diagnosticar problemas en las capas de presentación y de lógica de negocio existen se utilizan las siguientes herramientas:

HERRAMIENTAS DE RECOLECCIÓN DE DATOS Y DIAGNÓSTICO

- "IBM Thread and Monitor Dump Analyzer for Java". Esta herramienta permite analizar los hilos de ejecución de la Java Virtual Machine. Básicamente permite ver hilos bloqueados. Lo que se buscó en este caso particular fue el momento en que se consumieron todos los hilos del tipo WebContainer. Y se verificó en que método quedaban bloqueados o bien esperando otra operación. Este programa se alimenta se los archivos JavaCore generados en los Dumps de JVM. Para obtener el archivo a analizar hace falta ejecutar el comando kill -3 [PID]. A continuación se muestra un ejemplo en donde esta aplicación muestra un resumen del estado de los hilos de ejecución de Portal. Dicho resumen puede visualizarse a la derecha mientras que a la izquierda se puede acceder a información de cada uno de los hilos.

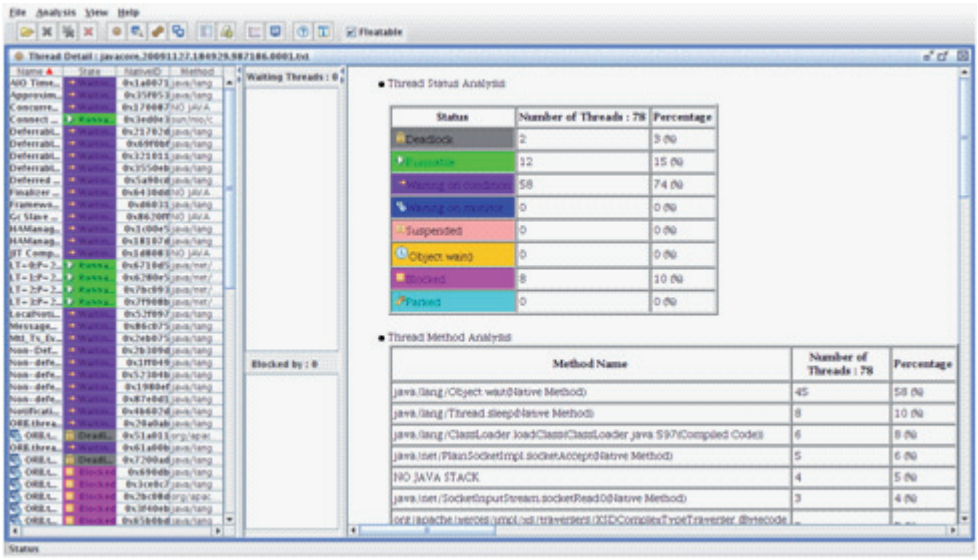


ILUSTRACIÓN XXIII - IBM THREAD AND MONITOR DUMP ANALYZER FOR JAVA

- “Visual Performance Analizer”: Esta herramienta permite analizar los resultados de comando “tprof”. Permite ver cuanta CPU consume un determinado proceso o hilo de ejecución. Luego utilizando el ID del hilo se puede obtener el stack de código que se está ejecutando utilizando la herramienta “IBM Thread and Monitor Dump Analyzer for Java”.

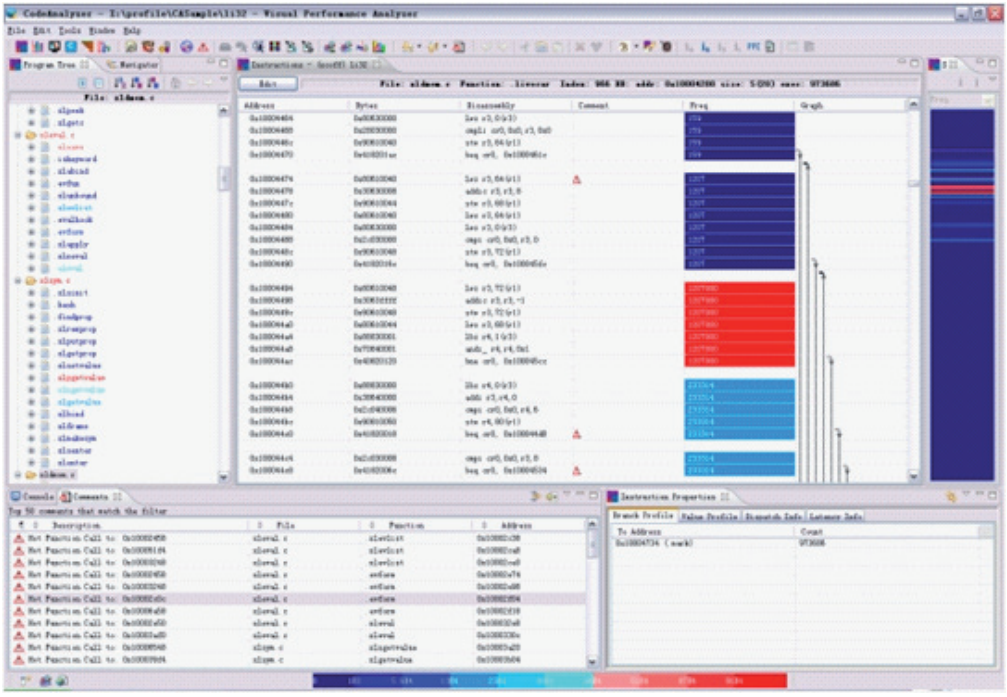


ILUSTRACIÓN XXIV - VISUAL PERFORMANCE ANALIZER

- “Garbage Collector Analizer”: Analiza la información de memoria y comportamiento del Gargabe Collertor de la JVM. Para que el Servidor de Aplicaciones guarde la información para analizar se requiere modificar parámetros en la Virtual Java Machine.

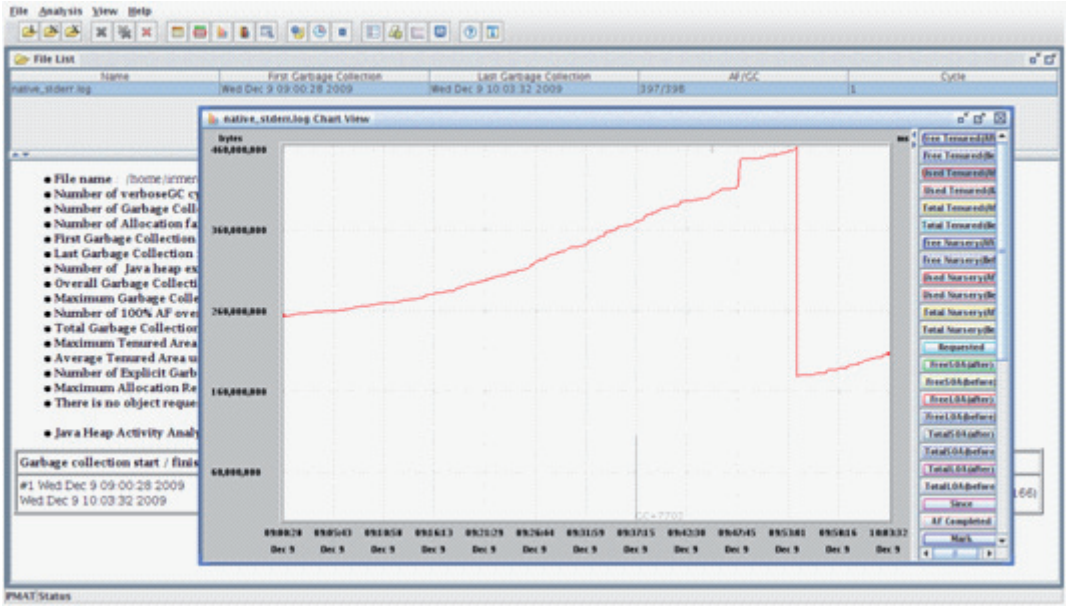


ILUSTRACIÓN XXV - GARBAGE COLLECTOR ANALIZER

- IBM Heap Analizer: Esta herramienta permite encontrar “Leaks” (Fugas) de Memoria que conllevan a un OutofMemory. Lo que se analiza con esta herramienta es un archivo que tiene un volcado del a memoria de la Java Virtual Machine.

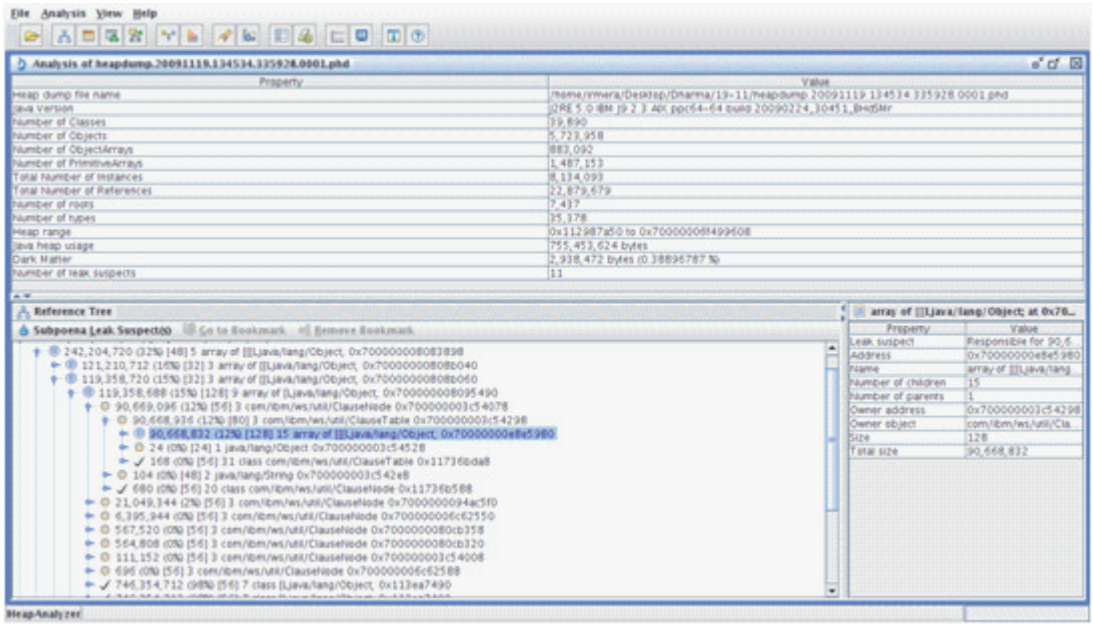


ILUSTRACIÓN XXVI - IBM HEAP ANALIZER

- Database Connection Pool Analyzer for IBM WebSphere Application Server: Esta herramienta permite analizar el comportamiento del pool de conexiones a la base de datos.

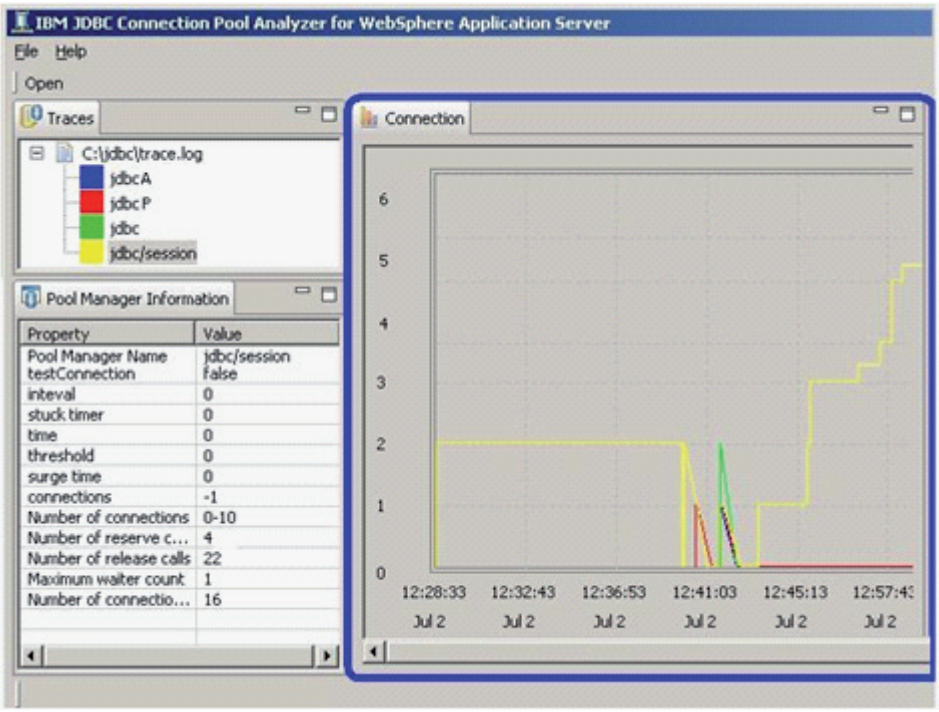


ILUSTRACIÓN XXVII - DATABASE CONNECTION POOL ANALYZER FOR IBM WEBSPHERE APPLICATION SERVER

PROBLEMAS MÁS COMUNES

Los problemas más comunes que se encuentran en este tipo de arquitecturas son los siguientes:

Problemas de Memoria:

- El primer problema que se encontró al momento de realizar las primeras pruebas sobre este Portal fueron errores del tipo OutOfMemory en el servidor de aplicaciones del contenedor de EJBs. El error anterior conllevaba a que los hilos del tipo WebContainer quedaran en espera de una respuesta

que nunca iba a llegar. Por lo cual al consumir el número máximo de hilos del tipo WebContainer el Application Server dejaba de responder y por consiguiente toda la solución fallaba.

Utilizando la herramienta “IBM Heap Analizer” se determinó que la aplicación estaba creando demasiados objetos del tipo EntityManagerFactory. Estos objetos eran muy pesados en memoria y provocaban un OutOfMemory en el Servidor. Para solucionar el problema se aplicó un patrón de diseño Singleton para que la aplicación reutilice el mismo objeto y no los cree de forma constante con cada solicitud.

- A largo plazo, luego de 2 o 3 días de vida, el Servidor de Aplicaciones dedicado a EJBs presentaba problemas de alta paginación generando tiempos de indisponibilidad.
  Primero se analizó con la herramienta “IBM Heap Analizer” para verificar si a largo plazo había un leak de memoria. Al no encontrarse nada se procedió a analizar el comportamiento del Garbage Collector. Se determinó que los tiempos de indisponibilidad fueron causados por ejecuciones del Garbage Collector. Para disminuir los tiempos de GC se bajaron los límites de Heap de Memoria de 2 GB a 512 MB. Esto causó que el GC corra varias veces al día en lugar de correr una única vez cada 3 días. Los ciclos de GC eran mucho más cortos. Solamente provocaban una indisponibilidad de menos de un segundo. Como segunda alternativa se está evaluando cambiar la política de GC.

Alto consumo de CPU:

En el Servidor de Aplicaciones del WebSphere Portal se detectaron amesetamientos de CPU. Es decir que la CPU se mantenía fija en un 60% o en un 90% de procesamiento luego de producirse un salto muy pronunciado.

Primero se analizó el problema con la herramienta “IBM Thread and Monitor Dump Analyzer for Java” pero era imposible determinar cual hilo que se encontraba en estado “running“era el que estaba causando el problema. Para determinar el o los hilos de ejecución que consumían marcadamente mucha CPU se utilizó la herramienta “Visual Performance Analizer”. Cuando la CPU del equipo se encontraba a 90% se pudo determinar que había 2 hilos idénticos que estaban consumiendo alrededor de 25% de CPU cada uno. Esta herramienta indica el ID de hilo con el cual se puede determinar que stack se está ejecutando. Los hilos en ejecución pertenecían a una sub-aplicación del Portal que hace uso de una herramienta denominada WebClipping. El problema estaba directamente asociado al producto Portal por lo cual fue derivado al laboratorio de IBM para su investigación. Aún el problema persiste.

Conexiones a la base de datos:

En el Servidor de Aplicaciones del WebSphere Portal se observó que momentáneamente existen bajas de performance en cuanto a tiempos de respuestas. Se focalizó la investigación en verificar que sucedía con los WebContainers en esos momentos particulares. Siguiendo el detalle de los hilos por medio de la herramienta “IBM Thread and Monitor Dump Analyzer for Java” se determinó que los hilos quedaban en estado “Wait” esperando la respuesta de la ejecución de un método de un EJB. Siguiendo el Stack Trace del EJB se determinó que el hilo quedaba en estado “Wait” esperando una conexión a la base de datos.

Por medio de la herramienta “Database Connection Pool Analyzer for IBM WebSphere Application Server” se logró mejorar la configuración del pool de conexiones para solucionar los problemas de conexión a la base.

V - DISCUSIÓN Y CONCLUSIONES

Luego del desarrollo las 3 arquitecturas y el ejemplo de un caso real y particular es inevitable que se presente la siguiente pregunta:

¿Qué modelo de arquitectura es el más adecuado para un nuevo proyecto de un Portal?

¿Se debería comenzar con la instalación del primer modelo de arquitectura y luego aplicar mejoras hasta alcanzar al modelo siguiente? ¿O bien ir directamente por un modelo específico más avanzado?

Para poder responder a esta pregunta es necesario analizar los modelos desarrollados utilizando los criterios de selección de topologías descriptos en el marco teórico de esta tesina.

A continuación, se resume en un cuadro comparativo, el cumplimiento de los criterios en cada una de las arquitecturas. A los criterios de selección también considero necesario agregar las variables de Costo de la Solución y tiempos de instalación. Dichas variables siempre se encuentran en un proyecto y que son prioritarias al momento de tomar una decisión en cualquier proyecto de sistemas.

Luego, en función de los requerimientos deseados del Portal y del cumplimiento de los criterios de selección se puede tomar una decisión aproximada.

	Arquitectura 1	Arquitectura 2	Arquitectura 3	Arquitectura caso práctico
SEGURIDAD	Todos los modelos de arquitecturas utilizaron el mismo esquema de seguridad por medio de las herramientas de Tivoli. Implementar un Portal sin un esquema de seguridad robusto presenta un riesgo para la empresa ya que expone a los sistemas de la misma a una red pública como Internet.			
RENDIMIENTO	Su rendimiento va a estar asociado a los recursos de CPU y memoria que tenga asignado el equipo en donde se encuentra la solución.	El rendimiento de la capa de presentación va a estar dado por la cantidad de nodos que posea la solución. El rendimiento se ve mejorado por la aplicación de Caches.	El rendimiento de las capas de presentación y Negocio va a estar dado por la cantidad de nodos que posea la solución. El rendimiento se ve mejorado por la aplicación de Caches.	Su rendimiento va a estar asociado a los recursos de CPU y memoria que tenga asignado el equipo en donde se encuentra la solución.
DISPONIBILIDAD	Ante el fallo de cualquiera de sus componentes la solución presentará indisponibilidades	Ante un fallo en la capa de presentación el modelo tiene redundancia para operar con otro nodo. Si existiera un fallo en el resto de la aplicación esta entraría en indisponibilidad.	Ante un fallo en la capa de presentación o de negocio, el modelo tiene redundancia para continuar operar.	Ante el fallo de cualquiera de sus componentes la solución presentará indisponibilidades.
MANTENIBILIDAD	Es el esquema más sencillo para mantener. Los deploys se desarrollan sobre un único Servidor de Aplicaciones.	Ya no existe un único servidor sobre el cual realizar deploys. Para instalar EJBs hace falta realizar deploys sobre 2 servidores. Aumenta la complejidad para analizar errores de la aplicación.	Todos los Deploy se unifican en los Deployments Managers correspondientes para EJB y para Presentación. Aumenta la complejidad para analizar errores de la aplicación.	Los deploys se desarrollan sobre un único Servidor de Aplicaciones.

ESCALABILIDAD	Su escalabilidad se encontrará relacionada a la capacidad que tenga el equipo en crecer en nivel de Procesamiento y memoria.	La solución es escalable en la capa de presentación y en la capa de negocio.	La solución es escalable en todos los componentes críticos. Proxy reverso, Presentación, EJB, Caches.	Su escalabilidad se encontrará relacionada a la capacidad que tenga el equipo en crecer en nivel de Procesamiento y memoria.
ESTADO DE SESIÓN	No hace falta ningún mecanismo extra para mantener el estado de la sesión ya que sólo existe una instancia de la solución.	Hace falta configurar el Load Balancer y Cache Proxy para que mantengan la sesión.	Es necesario elegir un método para mantener la sesión en caso de que falla algún componente. El método propuesto es la configuración de un servidor de manejo de sesiones.	No hace falta ningún mecanismo extra para mantener el estado de la sesión ya que sólo existe una instancia de la solución.
COSTO DE LA SOLUCIÓN	Al aumentar el número de servidores para escalar la capacidad de procesamiento y para proveer redundancia para alta disponibilidad el costo de la solución será más alto. En el caso particular de WebSphere Portal, su costo va a variar en base a la cantidad de procesadores utilizados en la solución.			Al ser una solución con poca cantidad de servidores y de productos es la solución menos costosa.
TIEMPO DE INSTALACIÓN	Al agregar más componentes a la solución (Load Balancers, Caches, Nodos, etc) los tiempos de instalación, configuración, pruebas y tunning aumentan.			Es la solución que menos tiempo de instalación requiere.

V.I - ¿Cuándo es conveniente aplicar el modelo 1?

Este modelo sólo es recomendable cuando se necesita implementar un modelo prototipo de un Portal ya que este modelo no es escalable ni presenta un esquema de alta disponibilidad.

No es necesario que la solución tenga un alto rendimiento ya que es un prototipo que posiblemente se utilice para analizar la viabilidad de un proyecto más grande.

Sólo se necesita un único servidor por lo cual representa un menor gasto para un proyecto.

Los tiempos de instalación y configuración de una solución del tipo “Servidor Único” son menores a los necesarios para instalar.

V.II -¿Cuándo es conveniente aplicar el modelo 2?

Esta es la solución mínima recomendable para que un Portal Web se implemente en producción. Si bien no posee alta disponibilidad en todos sus componentes su capa de presentación puede ser escalable por medio de la instalación de un nuevo nodo.

Al estar la capa de presentación y la capa de negocio instalados en diferentes Servidores de Aplicaciones permite el aislamiento de problemas. Por ejemplo si existirá una fuga de memoria en el contenedor de

EJB solamente dejaría fuera de servicio a dicha capa y no a la de presentación. Si bien las funcionalidades que llamen a lógica de negocio quedarían fuera de línea otras secciones del Portal que sólo utilicen el servidor de contenidos continuarían funcionando.

Al no tener alta disponibilidad en todos sus componentes no se recomienda utilizar esta solución para funcionalidades que sean el núcleo de una empresa.

V.III -¿Cuándo es conveniente aplicar el modelo 3?

Esta es una propuesta óptima y la más compleja para implementar un Portal Web en producción cuya cantidad de usuarios sea creciente tras el transcurso del tiempo.

Esta arquitectura permite un escalonamiento horizontal en sus componentes más importantes (Proxy reverso, Capa de Presentación y Capa de Lógica de Negocios).

Sus componentes claves están configurados bajo un esquema de alta disponibilidad para que la aplicación opere ante la ocurrencia de fallos. Es un escenario ideal para disponibilizar funcionalidades núcleo de una empresa.

Si bien es el modelo más completo, es la solución más costosa por la cantidad de servidores que se necesitan para tener componentes redundantes.

Su complejidad hará más difícil su mantenimiento y el análisis de incidencias productivas.

Esta solución puede alcanzarse de forma escalonada partiendo del modelo 2.

V.IV - Opinión sobre caso práctico real

La arquitectura expuesta en el caso práctico es muy similar a la planteada en el modelo 1. Su única diferencia es que el contenedor de EJBs se instaló en un servidor separado de la capa de presentación. Esta topología permite aislar problemas entre las capas de presentación y de negocio.

De acuerdo a lo investigado y analizado en esta tesina la arquitectura de la solución propuesta no es conveniente para un entorno productivo ya que la solución no es ni escalable ni es tolerante a fallos. Ante la ocurrencia de una falla en alguno de sus componentes toda la solución deja de estar disponible. Hoy en día sus mayores problemas se encuentran en el módulo de EJB ya que presenta caídas tras el transcurso de 72 horas aproximadamente.

En la actualidad solamente se presentan picos de 40 usuarios paralelos y no hay escases de recursos de Hardware. Una buena opción hubiera sido instalar un Cluster de Portal con un único nodo de manera tal que al momento de necesitar escalamiento horizontal o la posibilidad de alta disponibilidad se pudiera agregar un nuevo nodo (en el mismo equipo o en otro nuevo).

Tal como se encuentra en la actualidad esta arquitectura, cambiar a un esquema de solución en Cluster va a requerir de la re-instalación del WebSphere Portal. Para poder llevar a cabo esta acción sin interrumpir el servicio en Producción, será necesario realizar la instalación del Cluster en otro equipo y cuando el mismo se encuentre listo para ser productivo habrá que modificar las uniones de WebSeal para que derive los Request a dicho Cluster.

Conclusiones

En base a las arquitecturas propuestas en el desarrollo de esta tesina es viable construir una solución de Portal con requisitos mínimos y evolucionar la solución a medida que se sea necesario y que se tengan los recursos necesarios hasta alcanzar un sistema escalable, seguro, robusto y altamente disponible.

Ante el requerimiento de instalar una solución de Portal Web, se deberá analizar y seleccionar que características deberá cumplir la solución a corto y largo plazo con el objetivo de seleccionar la arquitectura que más se adecue a las necesidades a través del tiempo.

En el caso de que se requiera una rápida salida a producción la mejor opción es aplicar el primer modelo de arquitectura ya que es el modelo más sencillo de aplicar y el que tiene menor cantidad de requerimientos de hardware. Una vez en producción se deberá monitorear la carga de la solución para analizar en base a su carga si es necesario migrar a otro esquema de solución. En el caso de que sea necesario, se pueden instalar y configurar progresivamente nuevos componentes hasta llegar al modelo 3.

En el caso de que se requiera una solución que siempre se encuentre disponible y que sea tolerante a fallos, será necesario ir directamente al modelo de arquitectura 3 ya que todos sus componentes críticos se encuentran instalados de forma redundante. Los casos en donde se requiere este tipo de esquema son en aquellos en donde la solución web está orientada a ser núcleo de negocio de la empresa.

Es recomendable que la arquitectura seleccionada para las Capas de Presentación y Lógica de Negocio de un entorno Productivo pueda escalar de forma horizontal.

Los modelos presentados en esta tesina son propuestas, existen muchas formas de combinar los componentes de manera tal que se puedan alcanzar diferentes objetivos dependiendo de la naturaleza de las funcionalidades y de su importancia en el negocio de la organización.

VI - LÍNEAS FUTURAS DE INVESTIGACIÓN

Como líneas futuras de investigación se pueden desarrollar los siguientes trabajos:

- Análisis comparativos de rendimiento y funcionalidades entre diferentes productos que presten una solución de Portal.
- Diseñar una metodología para diagnóstico de problemas en arquitecturas complejas.
- Diseñar una metodología para análisis y diseño de funcionalidades para un Portal.
- Solución de Portales mediante la implementación de Portlets remotos utilizando el protocolo WSRP (**Web Services for Remote Portlets**)

VII - GLOSARIO

Concepto	Definición
Alta disponibilidad	Permite detectar la interrupción de un servicio y proporciona mecanismos de recuperación en caso de que se produzca un fallo en el sistema o un defecto en el proceso. Además, la alta disponibilidad permite a un sistema de copia de seguridad controlar los servicios en caso de que se haya producido un fallo en el sistema principal. También se conoce como “HA”. Fuente: Glosario de Sun Java Enterprise System
Annotations	Metainformación que se puede agregar a las clases, métodos y variables. Esta metadata permite evitar escribir código fuente ya que la herramienta será la encarga de procesar estos datos y generar el código necesario. Está orientado a la programación declarativa Fuente: http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html
API	(interfaz de programación de aplicaciones) Conjunto de instrucciones que un programa informático puede utilizar para comunicarse con otro software o hardware diseñado para interpretar dicha API. Conjunto de convenciones o instrucciones de llamada que define el modo en que los programas solicitan servicios en paquetes de software existentes. Fuente: Glosario de Sun Java Enterprise System
Aplicación multi-capa	Aplicación que divide las responsabilidades en N capas de Software. Fuente: Ejb 3 Developer Guide, Chapter 1, Pag 7
Bridging	Es una tecnologia que le permite a un Portlet generar contenidos con otras tecnologías como servlets, JSP y JSF. Fuente: Java Portlet Specification Version 1, Pag 16
Cache	Copia de datos originales que se almacena de forma local. Los datos almacenados en la memoria caché no tienen que ser recuperados de un servidor remoto cuando se solicitan. Fuente: Glosario de Sun Java Enterprise System
Ciclo de Vida	(1) (n.) Los eventos del marco de trabajo de la existencia de un componente de J2EE. Cada tipo de componente tiene eventos de definición que marcan su transición a estados en los que tiene disponibilidad de uso variable. Por ejemplo, se crea un servlet y su contenedor llama a su método init antes de la invocación de su método service por parte de clientes u otros servlets que requieran sus funciones. Después de la llamada de su método init, tiene los datos y la capacidad de preparación de uso para el que estaba diseñado. Su contenedor llama al método destroy del servlet antes de la finalización de su existencia, de forma que el procesamiento asociado al bobinado puede realizarse y pueden liberarse los recursos. Los métodos init y destroy de este ejemplo son callback method (método de retorno de llamada) s. Factores similares se aplican al ciclo de vida de todos los tipos de componentes de J2EE: enterprise beans, componentes web (servlets o páginas JSP), applets y clientes de aplicaciones. (2) (n.) Conjunto de fases durante el cual se recibe una solicitud de una página JavaServer Faces, se procesa un árbol de componente UI que representa la página procesada y se crea una respuesta. (3) (n.) Los eventos de marco de trabajo del tiempo de ejecución de un servidor, desde el inicio hasta el cierre, ambos incluidos. Fuente: Glosario de Sun Java Enterprise System
Cluster	Grupo de servidores, agentes o nodos conectados por medio de una red de alta velocidad que actúan como si fuera un único servidor, agente o nodo. Si falla uno de los servidores, agentes o nodos del clúster, sus servicios pueden conmutar por error a uno operativo. Fuente: Glosario de Sun Java Enterprise System

Contenedor	Proporciona servicios de administración del ciclo de vida, de seguridad, implementación y tiempo de ejecución a un tipo específico de componente de J2EE. El servidor de aplicaciones (Application Server) proporciona contenedores para todos los tipos de componentes de J2EE. Fuente: Glosario de Sun Java Enterprise System
CORBA	(arquitectura de agente de solicitudes de objetos) (n.)Definición de arquitectura estándar e independiente del lenguaje para informática distribuida orientada a objetos especificada por elOMG. Fuente: Glosario de Sun Java Enterprise System
DMZ	En seguridad informática, una zona desmilitarizada (DMZ, demilitarized zone) o red perimetral es una red local que se ubica entre la red interna de una organización y una red externa, generalmente Internet. Fuente: Glosario de Sun Java Enterprise System
Enterprise Bean	Componente de J2EE que implementa una tarea empresarial o una entidad empresarial y se encuentra alojada en un contenedor EJB; ya sea un entity bean (bean de entidad), message-driven bean (bean motivado por mensaje) o session bean. Fuente: Glosario de Sun Java Enterprise System
Entity Manager	Es un componente que provee servicios de persistencia, manejo de transacciones y manejo del ciclo de vida de las entidades. Fuente: Ejb 3 Developer Guide, Chapter 1, Pag 12
Firewall	También conocido como cortafuegos. Configuración de red, normalmente de hardware y software, que protege a los equipos de una organización que están conectados en red de un posible acceso exterior. Los servidores de seguridad se utilizan normalmente para proteger información, como un correo electrónico de la red o archivos de datos, dentro de un edificio físico o un sitio de la organización. Fuente: Glosario de Sun Java Enterprise System
Framework	“Un framewor es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.” Fuente: http://www.codebox.es/glosario
Garbage Collector	Es un mecanismo para liberar porciones de memoria que ya no son utilizadas. Fuente: The Java Virtual Machine Specification, Second Edition. Capitulo I
Heap	Porción de memoria que es compartida entre todos los hilos de ejecución de la JVM. Fuente: The Java Virtual Machine Specification, Second Edition. Capitulo I
HTTP	(protocolo de transferencia de hipertexto) (n.) Protocolo de Internet basado en Protocolo de control de transmisión/Protocolo de Internet que recupera objetos de hipertexto de host remotos. Los mensajes HTTP consisten en solicitudes del cliente al servidor y en respuestas del servidor al cliente. Fuente: Glosario de Sun Java Enterprise System
Java Content Repository	Es una especificación Java que unifica la forma de acceder a un repositorio de contenidos. Fuente: Content Repository API for Java™ Technology Specification.
Java EE	Entorno para desarrollar e implementar aplicaciones empresariales basadas en web y con varias capas. La plataforma J2EE consiste en un conjunto de servicios, API y protocolos que proporcionan las funciones necesarias para desarrollar estas aplicaciones. Fuente: Glosario de Sun Java Enterprise System.

Javascript	Lenguaje de secuencias de comandos compacto basado en objetos que se utiliza para desarrollar aplicaciones de Internet de cliente y de servidor. Fuente: Glosario de Sun Java Enterprise System
JNDI	(nombre de JavaNaming and Directory Interface) Nombre utilizado para acceder a un recurso que se ha registrado en el servicio de nombres JNDI. Fuente: Glosario de Sun Java Enterprise System.
JPA	Java Persistence API. Es un motor de persistencia que permite interactuar a la capa de negocio con la capa de persistencia Fuente: Ejb 3 Developer Guide, Chapter 1, Pag 12.
JPQL	Java Persistence Query Language. Es un lenguaje de consulta que permite recuperar información de una base de datos relacional. Fuente: Ejb 3 Developer Guide, Chapter 1, Pag 12.
JSF	Marco de trabajo para la creación de interfaces de usuario en el servidor para aplicaciones web escritas en el lenguaje de programación Java. Fuente: Glosario de Sun Java Enterprise System
JSP	Tecnología web extensible que utiliza datos estáticos, elementos JSP y objetos Java de servidor para generar contenido dinámico para un cliente. Normalmente, los datos estáticos son elementos HTML o XML y, en muchos casos, el cliente es un navegador web. Las páginas creadas con tecnología JSP combinan las capacidades de diseño de una página de navegador estándar con la potencia de un lenguaje de programación. Extensiones que permiten realizar todas las metafunciones de la tecnología JSP, como instalación, inicialización, destrucción, acceso desde otros componentes y gestión de la configuración. Aplicaciones Java reutilizables que se ejecutan en un servidor web en vez de en un navegador web. Fuente: Glosario de Sun Java Enterprise System
JTA	(API de transacción Java) (n.) Una API que permite a las aplicaciones y servidores J2EE acceder a transacciones. Fuente: Glosario de Sun Java Enterprise System
JVM	Java Virtual Machine. Son un conjunto de rutinas de código nativo (especifico para cada plataforma) que tiene como objetivo interpretar y ejecutar instrucciones que se encuentren en ByteCode. Fuente: The Java Virtual Machine Specification, Second Edition. Capitulo I
LDAP	(Protocolo ligero de acceso a directorio) (n.) Protocolo de servicio de directorios diseñado para ejecutarse mediante TCP/IP y en numerosas plataformas. Simplificación del protocolo de acceso al directorio (DAP) X.500 que permite un único punto de administración para almacenar, recuperar y distribuir información, incluidos los perfiles de los usuarios, las listas de distribución y los datos de configuración por todos los servidores de Sun Java System. Directory Server utiliza el protocolo LDAP. Fuente: Glosario de Sun Java Enterprise System
MDB	(bean controlado por mensajes) (n.) Enterprise Bean que es un consumidor de mensajes asíncrono. Un Bean controlado por mensajes no posee estado para un cliente específico, sino que sus variables de instancia pueden contener estado a través de la administración de los mensajes de clientes, incluida una conexión de base de datos abierta y una referencia de objeto a un objeto basado en la arquitectura EJBTM. Un cliente accede a un Bean controlado por mensajes mediante Fuente: Glosario de Sun Java Enterprise System

Node Agent	Agente ligero que se exige en cada máquina que aloja al menos una server instance (instancia de servidor) de Application Server. El agente de nodo realiza diferentes tareas, incluidas el inicio, la parada, la creación y la eliminación de instancias de Application Server. Fuente: Glosario de Sun Java Enterprise System.
Nodo	Nodo informático. Uno de los distintos equipos de un entorno de red o de Internet. Las aplicaciones distribuidas se implementan en este entorno, con varios componentes distribuidos, servicios de negocios y servidores ejecutándose en varios nodos informáticos. Consulte también clúster. Fuente: Glosario de Sun Java Enterprise System
Object/Relational Mapping	Capacidad para vincular un modelo orientado a objetos a un modelo relacional de datos, normalmente el esquema de una base de datos relacional. Proceso que consiste en convertir un esquema en una estructura diferente. Fuente: Glosario de Sun Java Enterprise System
OMG	(Grupo de administración de objetos) (n.) Consorcio que produce y mantiene especificaciones del sector informático para aplicaciones empresariales interoperables. Su sitio web es http://www.omg.org/. Fuente: Glosario de Sun Java Enterprise System
ORB	(Grupo de administración de objetos) (n.) Consorcio que produce y mantiene especificaciones del sector informático para aplicaciones empresariales interoperables. Su sitio web es http://www.omg.org/. Fuente: Glosario de Sun Java Enterprise System
Persistence Manager	El responsable del administrador de la persistencia de un entity bean (bean de entidad) Fuente: Glosario de Sun Java Enterprise System
Persistencia	Para componentes, el protocolo de transferencia del estado entre las variables de la instancia y la base de datos subyacente. Fuente: Glosario de Sun Java Enterprise System
Portlet	Un Porlet se lo define como un componente Web basado en tecnología Java que se encarga de procesar solicitudes de usuarios y de generar contenido de forma dinámica. Fuente: Java Portlet Specification Version 1, Pag 13
Proxy reverso	Proxy que realiza tareas de reescritura y traducción bidireccional de direcciones URL entre clientes y servidores. A diferencia de un proxy, que se encuentra en el cliente, un proxy inverso reside en el servidor de la red. Fuente: Glosario de Sun Java Enterprise System
Rendering (Renderizacion)	Se lo puede definir como un proceso que se lleva a cabo para presentar una página Web. Por ejemplo Java Server Faces proporciona mecanismos para generar código HTML y así presentar una página Web. Fuente: Glosario de Sun Java Enterprise System.
Request (HTTP)	De acuerdo al Consorcio de la World Wide Web (W3C), un Request define una operación a ser llevada a cabo en una URL. Fuente: http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
RMI	(llamada a métodos remotos) (n.) Tecnología que permite a un objeto que se encuentra en ejecución en una máquina virtual de Java invocar métodos que se estén ejecutando en otra máquina virtual de Java. Fuente: Glosario de Sun Java Enterprise System
RMI-IIOP	Versión de RMI implementada para utilizar el protocolo CORBA IIOP. RMI a través de IIOP permite la interoperabilidad con objetos CORBA implementados en cualquier lenguaje si se encuentran definidas originalmente todas las interfaces remotas como interfaces RMI. Fuente: Glosario de Sun Java Enterprise System

Servidor de Autoría	Instancia se utilizará para crear o modificar contenido web que luego será migrado al servidor de delivery a través de un mecanismo de migración denominado “Sindicación”. Fuente: Websphere Portal 6.1.0.1 Infocenter
Servidor de Delivery	Instancia que servirá para presentar el sitio Web al usuario final Fuente: Fuente: Websphere Portal 6.1.0.1 Infocenter
Servlet	Programa de servidor escrito en el lenguaje de programación JavaTM que amplía la funcionalidad de un servidor web, generando contenido dinámico e interactuando con aplicaciones web utilizando el paradigma solicitud-respuesta. Los servlets son parecidos a los applets en el sentido de que se ejecutan en el servidor. Sin embargo, los servlets no utilizan una interfaz de usuario. Fuente: Glosario de Sun Java Enterprise System
Single Sing On	Función que permite que la autenticación de un usuario en un servicio de un sistema distribuido se aplique automáticamente a otros servicios del sistema. Fuente: Glosario de Sun Java Enterprise System
SMS	Servidor de Manejo de Sesiones. Es un servidor que se utiliza para que las instancias de WebSeal persistan la información de las sesiones en el Cluster de Portal Fuente: Tivoli Infocenter
SOA	(arquitectura orientada a servicios) (n.)Describe una aplicación compleja compuesta por consumidores y proveedores de servicios. Los consumidores y proveedores pueden intercambiar mensajes sin referencia a la ubicación concreta de otro. La arquitectura aisla también los procesos principales de una aplicación de otros proveedores y consumidores de servicios. Fuente: Glosario de Sun Java Enterprise System
SSL	(capa de socket segura) (n.) Una forma de cifrado segura de bajo nivel que utilizan otros protocolos como HTTP y FTP. El protocolo SSL incluye disposiciones para la autenticación del servidor, el cifrado de datos en tránsito y la autenticación de cliente opcional. El protocolo permite que las aplicaciones cliente-servidor se comuniquen de una forma que no puede captarse ni modificarse. Fuente: Glosario de Sun Java Enterprise System
Thread safe	(Seguridad en hilos). Un objeto se considera Thread-Safe si puede ser invocado desde múltiples hilos de ejecución y no existe ningún tipo de interferencia entre dichos hilos. Fuente: http://www.ibm.com/developerworks/java/library/j-jtp09263.html
WebContainer	Un contenedor que implementa el contrato de componente web de la arquitectura J2EE. Este contrato especifica un entorno de tiempo de ejecución para componentes web que incluye servicios de seguridad, concurrencia, administración del ciclo de vida, transacción, implementación y otros servicios. Un contenedor web proporciona los mismos servicios como contenedor JSP así como vista federada de las API de la plataforma J2EE. Un contenedor web lo proporciona una web o un servidor J2EE. Fuente: Glosario de Sun Java Enterprise System
WSDL	(lenguaje de descripción de servicios web) (n.) Un lenguaje basado en XML utilizado para definir los servicios web de forma estándar.Describe tres propiedades básicas de un servicio web: definición del servicio web, el modo de acceder a ese servicio y su ubicación. Fuente: Glosario de Sun Java Enterprise System

VIII - BIBLIOGRAFÍA

Libros

- Anton Polgar, Robert Mark Bram. *Building and managing enterprise-wide portals*. (2006)
- Cal Henderson. *Building scalable web sites*. (2006). O'Really.
- Greg Barish. ***Building scalable and high-performance Java Web applications using J2EE***. (2002) Person Education.
- Inderjeet Singh, Beth Stearns, Mark Johnson, and the Enterprise Team (2002). Designing *Enterprise Applications with the J2EETM Platform*, Second Edition. Addison-Wesley.
- Micheal Sikora. *EJB Developer Guide*. Packt Publishing (2008)

Manuales

- Axel Buecker, Ana Veronica Carreno, Norman Field, Christopher Hockings, Daniel Kawer, Sujit Mohanty, Guilherme Monteiro. *Enterprise Security Architecture Using IBM Tivoli Security Solutions*. (2007). IBM Reedbooks.
- Axel Buecker, Anji Greene. *Deployment Guide Series: IBM Tivoli Access Manager for e-business V6.0.* (2008). IBM Red Books.
- Axel Buecker, Chris Eric Friell, Armando Lemos, Rick McCarty, Jani Perttilä, Dieter Riexinger, Andreas Schmengler. *Enterprise Business Portals with IBM Tivoli Access Manager*. (2002). IBM Reedbooks.
- Birgit Roehm Gang Chen , Andre de Oliveira Fernandes , Cristiane Ferreira , Rodney Krick , Denis Ley, Robert R. Peterson , Gerhard Poul , Joshua Wong , Ruibin Zhou . *WebSphere Application Server V6 Scalability and Performance Handbook*. (2005). IBM RedBooks.
- *Concepts, Planning, and Installation for Edge Components Version 6.1* (2006)
- *Glosario de Sun Java Enterprise System*. (Marzo 2007)
- *IBM HTTP Server*, Version 6.1 – User Guide (2006)
- *IBM Tivoli Access Manager for Operating Systems Release Notes Version 5.1.* (2003)
- *IBM Tivoli Access Manager Installation Guide Version 5.1* (2003)
- *IBM WebSphere Portal 6.1.X Performance Tuning Guide* (2009) IBM
- Jerry Dancy, Ravinder Dhaliwal , Krishnan Hariharan , Barbara Koch , Greg Presayzen , Meredith Spitalnik , Hunter Tweed , Sherwood Yao . *WebSphere Portal Version 6 Enterprise Scale Deployment Best Practices*. (2007) IBM RedBooks
- John Ganci, Hinrich Boog, Melanie Fletcher, Brett Gordon, Ashwin Manekar. *Develop and Deploy a Secure Portal Solution*. (2004). IBM RedBooks.
- Ken Ueno, Tom Alcott , Jeff Blight , Jeff Dekelver, Daniel Julin, Colin Pfannkuch, Tzueing Shieh. *WebSphere Scalability: WLM and Clustering*. (2004). IBM RedBooks.
- Ueli Wahli, Giuseppe Bottura, Jacek Laskowski, Nidhi Singh. *WebSphere Application Server Version 6.1 Feature Pack for EJB 3.0* (2008).
- Ueli Wahli, Owen Burroughs , Owen Cline , Alec Go , Larry Tung. *Web Services Handbook for WebSphere Application Server Version 6.1* (2006). IBM RedBooks.

Papers

- Alex Louwe Kooijmans , Dinkar Tiwari. *Security with IBM Tivoli Access Manager V6 and IBM WebSphere Application Server V6 on IBM z/OS*. (2006)
- Bob Bretl, Allen Otis, Marc San Soucie, Bruce Schuchardt, R. Venkatesh. *Persistent Java Objects in 3 tier architectures* (2009)
- Channu Kambalyal. 3-Tier Architecture (2008)
- Hunter Tweed. *A Step-By-Step Guide to Configuring a WebSphere Portal v6.1.0.0 Cluster using WebSphere Application Server v6.1.0.15.* (2008)
- Michele Galic, Christian Pedersen, Oliver Trabert. *Portal Application Design and Development Guidelines* (2003)

Páginas Web

- IBM Tivoli Access Manager for e-business 6.1 Infocenter, http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame.doc/welcome.htm (Octubre 2009)
- IBM Webphere Portal 6.1 Infocenter, http://publib.boulder.ibm.com/infocenter/wpdoc/v6r1/index.jsp?topic=/com.ibm.wp.ent.doc_v615/welcome_main.html (Octubre 2009)
- Java theory and practice: Characterizing thread safety. http://www.ibm.com/developerworks/java/library/j-jtp09263.html (Mayo 2010)

- http://www.codebox.es/glosario (Mayo 2010)
- http://www.w3.org/Protocols/rfc2616/rfc2616.html (Junio)

Especificaciones
- Alejandro Abdelnur, Stefan Hepper. *JavaTM Portlet Specification Version 1.0*
- Linda DeMichiel. JSR 220: *Enterprise JavaBeansTM,Version 3.0 EJB Core Contracts and Requirements* (2006)
- Tim Lindholm, Frank Yellin. *The JavaTM Virtual Machine Specification* - Second Edition
- Content Repository API for Java™ Technology Specification. Version 1.0 (2005)

ANEXO I: DESCRIPCIÓN DE COMPONENTES DE IBM

IBM Tivoli Directory Server[24]

IBM Tivoli Directory Server es la implementación de IBM de un Lightweight Directory Access Protocol (LDAP) para los sistemas operativos Windows, AIX, Linux, Solaris y HP-UX. IBM Tivoli Directory Server se compone de los siguientes componentes:

- Una base de datos DB2 para almacenar la información del directorio
- Servidor Proxy que se encarga de routear operaciones del LDAP a otros servidores
- Una herramienta cliente
- Una interfaz gráfica para administrar servidores
- Una interfaz gráfica para administrar usuarios

IBM Tivoli Access Manager for eBusiness (TAM)[25]

TAM es una solución corporativa de seguridad que se encarga de proveer servicios de autenticación y autorización a nivel Web. Su objetivo es controlar el acceso de usuarios a información y recursos restringidos. Es una solución de control de acceso centralizada, flexible y escalable que permite construir infraestructuras seguras.

El Access Manager está conformado principalmente por 2 componentes núcleo fundamentales:

- Registro de Usuarios. Access Manager requiere de este registro para proveer funciones de autorización. Provee específicamente de:
  - Una base de datos de las identidades de los usuarios que son reconocidos por el Access Manager
  - Una representación de grupos en el Access Manager (Roles) que pueden ser asociados a usuarios
  - Un Almacén de datos de otros metadatos requeridos para soportar las funciones de autorización
- Servicio de Autenticación. Que a su vez está compuesto por una base de autorizaciones y un motor de autorización. Separadamente del registro de usuarios, el Access Manager utiliza una base de datos especial que contiene una representación virtual de los recursos que protege denominado “protected object space”. El mismo tiene un formato propietario y contiene definiciones que pueden representar recursos lógicos y físicos. La política de seguridad para estos recursos se puede implementar por medio de los siguiente mecanismos:
  - Access control list (ACL): Son objetos especiales que definen políticas tipos de usuarios que pueden ser considerados para acceder a cierto objeto. Especificando también el tipo de operación que le es permitida.

[24] Fuente: Axel Buecker, Ana Veronica Carreno, Norman Field, Christopher Hockings, Daniel Kawer, Sujit Mohanty, Guilherme Monteiro. Enterprise Security Architecture Using IBM Tivoli Security Solutions. (2007). IBM Reedbooks. (Paginas 72-80)
[25] Fuentes: Axel Buecker, Ana Veronica Carreno, Norman Field, Christopher Hockings, Daniel Kawer, Sujit Mohanty, Guilherme Monteiro. Enterprise Security Architecture Using IBM Tivoli Security Solutions. (2007). IBM Reedbooks. (Paginas 164-170)
Axel Buecker, Chris Eric Friell, Armando Lemos, Rick McCarty, Jani Perttilä, Dieter Riexinger, Andreas Schmengler. Enterprise Business Portals with IBM Tivoli Access Manager. (2002). IBM Reedbooks. (Paginas 4 - 14)
IBM Tivoli Access Manager for Operating Systems Release Notes Version 5.1. (2003)
IBM Tivoli Access Manager Installation Guide Version 5.1 (2003)

- Protected object policy (POP): Especifica condiciones adicionales que gobiernan el acceso protegido al recurso. Como privacidad, integridad, auditoría y tiempo del día de acceso.
- Extended attributes: son valores adicionales en una ACL o POP que pueden ser leídas e interpretadas por aplicaciones de terceros.

Estos componentes conforman el núcleo de TAM que llevan a cabo el siguiente conjunto de operaciones:

- Conocer la identidad de quien está llevando a cabo una operación particular (Usuario)
- Conocer los roles asociados a una identidad particular (Grupo)
- Conocer que identidad puede acceder a un determinado objeto o aplicación (Objetos)
- Conocer las reglas de autorización asociadas con objetos de aplicación (Políticas)
- Utilizar los conocimientos antes descriptos para tomar decisiones en nombre de las aplicaciones (Autorizaciones)
  - Auditar y llevar trazas de toda la actividad relacionada a autenticaciones y autorizaciones

TAM también está compuesto por los siguientes componentes que le dan al producto capacidades administrativas para controlar su función:

- Policy Server: Provee la administración de la base de datos de autorización y su distribución a los servicios de Autorización.
- Utilidad PDADMIN: Provee funcionalidades administrativas por medio de línea de comando.
- Web Portal Manager: Provee una interfaz Web con las mismas funcionalidades que el comando PDADMIN.

REQUERIMIENTOS DE HARDWARE (AIX)

- IBM POWER family of processors
- Requerimientos de espacio en Disco
  - 88 MB para el directorio /opt
  - 67 MB para el directorio /usr
  - 100 MB para el directorio /var
- Mínimo de 256 mb de memoria física recomendada
- Lectora CD-ROM

WebSEAL[26]

WebSEAL es un producto de la familia de Tivoli que provee servicios de Proxy Reverso. Se encuentra altamente relacionado con TAM. Entre sus principales características y funciones se encuentran:

- Una de las funciones clave de WebSEAL es la de proteger el acceso a contenidos web. Para ello, utiliza el servicio de Autorización del Access Manager. Este servicio debe saber qué objetos Web (es decir, cuales direcciones URL) requieren protección, y qué nivel de acceso a estos objetos se permite a los usuarios y grupos definidos en el registro de usuarios.Proveer de mecanismos de Single Sign-On
- Incorporar aplicaciones Servidores de Aplicaciones Web a las políticas de seguridad
- Da soporte a puerto HTTP y HTTPs
- Se encuentra implementado sobre la base de un HTTP Server

[26] Axel Buecker, Chris Eric Friell, Armando Lemos, Rick McCarty, Jani Perttilä, Dieter Riexinger, Andreas Schmengler. Enterprise Business Portals with IBM Tivoli Access Manager. (2002). IBM Reedbooks. (Paginas 14 - 24)

A continuación se ilustra la arquitectura integrada entre TAM y WebSEAL

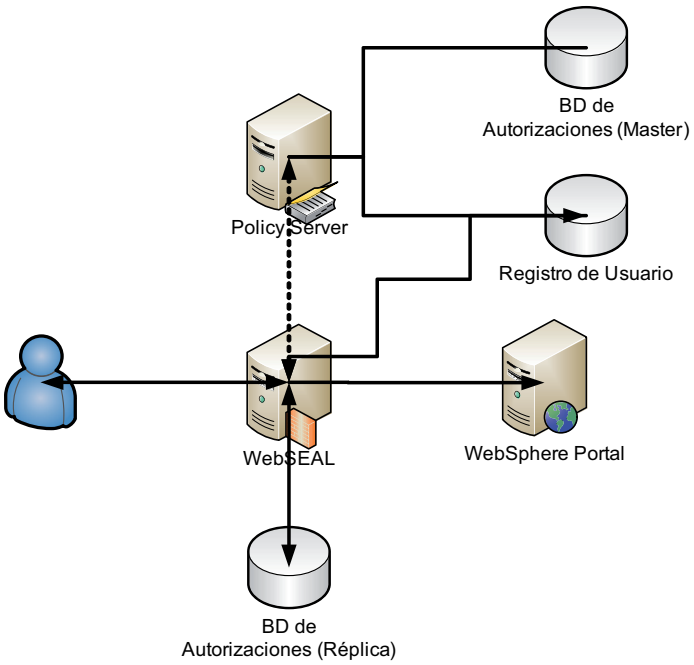


ILUSTRACIÓN XXVIII - ARQUITECTURA TAM Y WEBSEAL

Los pasos para la Autenticación y la Autorización son los siguientes:

1. WebSEAL autentica el usuario utilizando el Registro de Usuarios
2. Una vez el usuario autenticado WebSEAL, crea las credenciales necesarias
3. Haciendo uso de la réplica de la BD de Autorizaciones WebSeal verificar la autorización del usuario a acceder al recurso.
4. Si el usuario pasa la autorización se le permite acceder al objeto.

REQUERIMIENTOS DE HARDWARE (AIX)

- IBM POWER family of processors
- Requerimientos de espacio recomendado en Disco 100 mb
- Mínimo de 256 MB de memoria física recomendada
- Lectora CD-ROM

WebSphere Application Server

WebSphere Application Server (WAS) es la implementación de IBM de la plataforma Java 2 Enterprise Edition. WAS puede ejecutar los siguientes tipos de aplicaciones:

- J2EE Aplicaciones: La especificación Java 2 Platform Enterprise Edition es un estándar para desarrollar, deployar y ejecutar Aplicaciones Empresariales. WAS 6.1 provee un soporte completo para la especiaficación J2EE.
- Porlets Applications: El WAS posee de un contenedor de Portlets que provee un entorno de ejecución de Portlet que obedezcan la especificación JSR 168.
- Aplicaciones SIP (Session Initiation Protocol): Son programas que utilizan al menos un servlet del tipo SIPque se encuentra escrito bajo la norma JSR116. SIP se utiliza para establecer, modificar y finalizar sesiones IP multimediales.

En otras palabras IBM WebSphere Application Server es un servidor de aplicaciones web que proporciona servicios J2EE al entorno de WebSphere Portal. Ejecuta los Portlets de Java, JavaServer Pages (JSP), JavaBeans y Enterprise JavaBeans (EJB) utilizados por el Portal.

REQUERIMIENTOS DE HARDWARE (AIX)

- IBM POWER family of processors
- Requerimientos de espacio en Disco
  - 930 MB para /usr/IBM/WebSphere/AppServer
  - 100 MB para el directorio /tmp
- Mínimo de 1 GB de memoria física recomendada
- Lectora CD-ROM

WebSphere Portal[27]

WebSphere Portal es una aplicación J2EE que se ejecuta sobre WebSphere Application Server. Su principal función consiste en servir de infraestructura de renderizado para los usuarios del portal que realicen alguna solicitud dentro del Portal Web. WebSphere Portal crea un entorno que proporciona los servicios de conectividad, administración y presentación necesarios. WebSphere Portal incluye varias funciones y mejoras nuevas que facilitan el diseño, la administración y la utilización del portal. De acuerdo al sitio Oficial del producto (Infocenter) sus principales funciones son[28]:

- "Infraestructura versátil: IBM WebSphere Portal proporciona a los usuarios una vista coherente de las aplicaciones del portal y permite a los usuarios definir conjuntos específicos de aplicaciones que se presentan en un contexto único. Dependiendo del dispositivo solicitante, la generación de este conjunto de aplicaciones variará para cumplir los requisitos del dispositivo."
- "Personalización: Personalizar la experiencia del usuario es uno de los objetivos principales de IBM WebSphere Portal. Se proporcionan portlets de administración y de usuario para personalizar el contenido y el aspecto y diseño de las páginas. Además, se facilitan herramientas que permiten a los expertos en la materia personalizar el contenido según las necesidades e intereses de cada visitante del sitio."
- "Gestión de contenido: IBM Lotus Web Content Management y Portal Personalization proveen funcionalidad para crear y gestionar contenido para el portal o el sitio web. Estas aplicaciones se utilizan por separado o conjuntamente como una solución de contenido integrada para el portal o el sitio web."
- "Portlets: Los portlets son una parte importante de IBM WebSphere Portal. Los portlets son pequeñas aplicaciones que se desarrollan, despliegan, gestionan y visualizan de forma independiente. Los administradores y los usuarios componen páginas personalizadas seleccionando y ordenando los portlets, lo que da como resultado páginas web personalizadas."
- "Integración de aplicaciones: Un portal proporciona acceso al contenido, los datos, y los servicios localizados en toda la empresa. Estos servicios no sólo incluyen los conectores y portlets predefinidos sino también las herramientas para crear conectores y portlets adicionales."
- "Colaboración y Funciones clave de búsqueda de Portal"

REQUERIMIENTOS DE HARDWARE (AIX)

- Mínimo de 2.5 GB de espacio en disco para instalar WebSphere Portal
- Mínimo de 1 GB de espacio en disco para instalar WebSphere Application Server
- Se recomienda un mínimo de 4 GB para una instalación completa
- Para una carga mínima WebSphere Portal funciona con 2 GB de RAM. Sin embargo lo optimo son 4 GB
- Lectora CD-ROM

[27] Fuente: Websphere portal 6.1 Infocenter
[28] Funciones Citadas del sitio WebSphere Portal Infocenter 6.1 (http://publib.boulder.ibm.com/infocenter/wpdoc/v6r1/topic/com.ibm.wp.ent.doc_v6101/overview/intr_fea.html)

IBM HTTP Server[29]

HTTP Server es un servidor web que se puede utilizar como base de cualquier aplicación de e-business. El nuevo software de e-business de IBM, como la familia de productos WebSphere, se ha diseñado para operar con muchos servidores web populares. Entre las funciones de IBM HTTP Server, se incluyen:

- Soporte a las conexiones seguras SSL

- Fast Response Cache Accelerator
- Soporte de IBM como parte del paquete WebSphere
- Soporte a la criptografía de hardware
- Servidor de administración que ayuda a administrar y configurar los servidores IHS

Caching Prox[30]

Este componente de IBM reduce la utilización del ancho de banda y mejora la velocidad de carga de una página web. Este componente se encarga de guardar en memoria y de proveer contenidos estáticos que se utilicen en el Portal. Ante una solicitud de un usuario, este componente solicita el contenido al servidor de aplicaciones y se encarga de devolver la respuesta. Posteriormente se encarga de guardar el contenido en un cache local de manera tal que ante una misma solicitud el contenido pueda ser recuperado desde el cache en lugar de que el servidor de aplicaciones tenga que resolver nuevamente el contenido.

Load Balance[31]

Es un componente que se encarga de administrar el flujo de trafico de red reduciendo congestiones y balaceando la carga entre los diferentes componentes y equipos. Su función consta en capturar las solicitudes y redireccionarlas al servidor de aplicaciones que tenga sea más apto para poder ejecutar lo solicitado. En otras palabras balancea la carga de trabajo entrante entre un conjunto definido de servidores capaces de resolver la misma tarea.

Dispatcher[32]

Es un componente que para todos los servicios de internet como HTTP, FTP, HTTPs y Telnet provee mecanismos para realizar un balance de carga entre servidores dentro de una LAN o de una WAN. Para servicios HTTP, este componente puede hacer balanceo de carga basándose en la URL solicitada por un usuario.

[29] Fuente: IBM HTTP Server, Version 6.1 – User Guide (2006)
[30] Fuente: Concepts, Planning, and Installation for Edge Components Version 6.1 (2006)
[31] Fuente: Concepts, Planning, and Installation for Edge Components Version 6.1 (2006)
[32] Fuente: Concepts, Planning, and Installation for Edge Components Version 6.1 (2006)

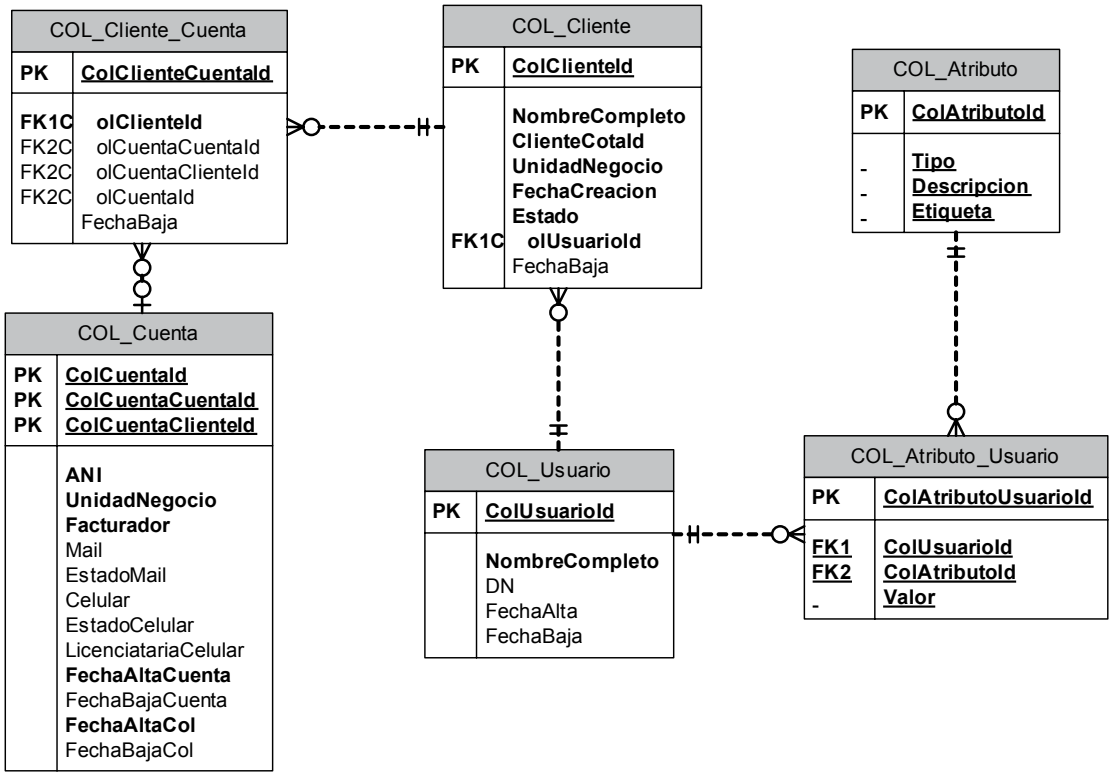
ANEXO II: DISEÑO TÉCNICO DE FUNCIONALIDAD DE VISUALIZACIÓN DE DATOS DE FACTURAS

El presente anexo contiene el diseño técnico correspondiente a la funcionalidad para visualizar imágenes de las facturas.

Estructura física de la información

Código Objeto/Relación (Especificación Técnica)	Nombre del Objeto	Descripción del Objeto
OB-DT001-001	**COL_Usuario**	Tabla que contiene información básica de Usuario.
OB-DT001-002	**COL_Cuentas**	Tabla que contiene las cuentas que un usuario registró por medio de un CPE.
OB-DT001-003	**COL_Atributo**	Tabla que contiene atributos asociables.
OB-DT001-004	**COL_Atributo_Usuario**	Tabla que contiene los valores de los atributos asociados a un Usuario.
OB-DT001-005	**COL_Cliente**	Tabla que contiene los Clientes asociados a un usuario

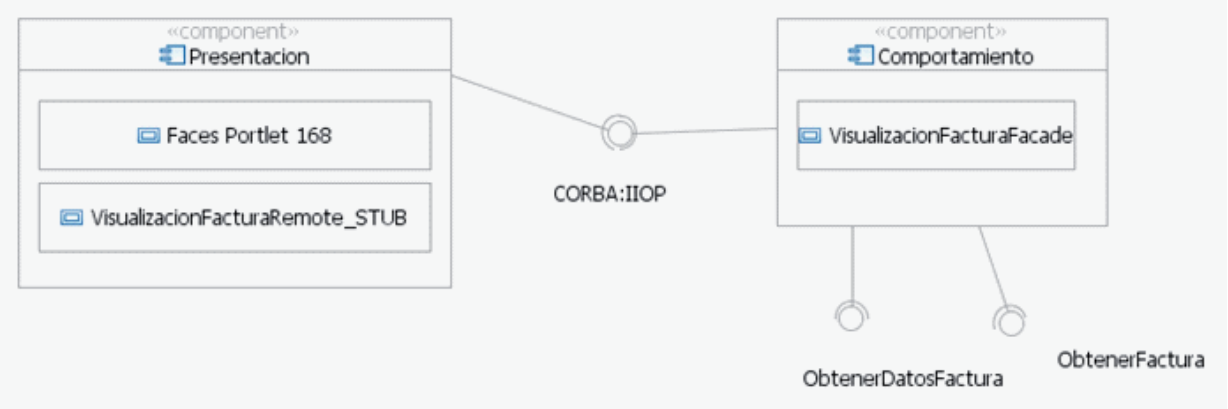
Modelo de datos



Especificación de componentes software

Código Componente Sw	Nombre del Componente Sw	Descripción del Componente
COM-DT001-001	**Comportamiento**	Capa que contiene la lógica de negocio y la conexión con la fuente de datos.
COM-DT001-002	**Presentación**	Capa que contiene la lógica para generar la presentación de la aplicación Web.

Diagrama de componentes

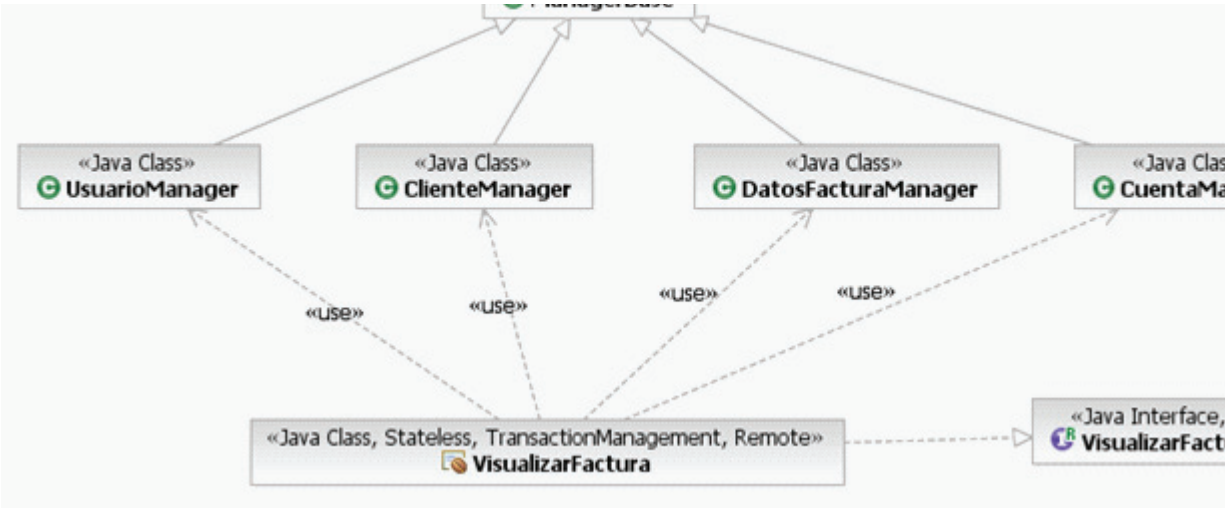


Descripción de componentes

Componente COM-DT001-001 (Comportamiento)

Descripción del componente Sw COM-DT001-001	
Nombre	Capa de comportamiento
Versión	1.0
Tipo	Componente EJB 3
Descripción	Este componente contendrá la lógica de negocio necesaria para recuperar y mostrar los datos de las facturas de una cuenta.

DIAGRAMA DE CLASES (FACADE/MANAGERS)

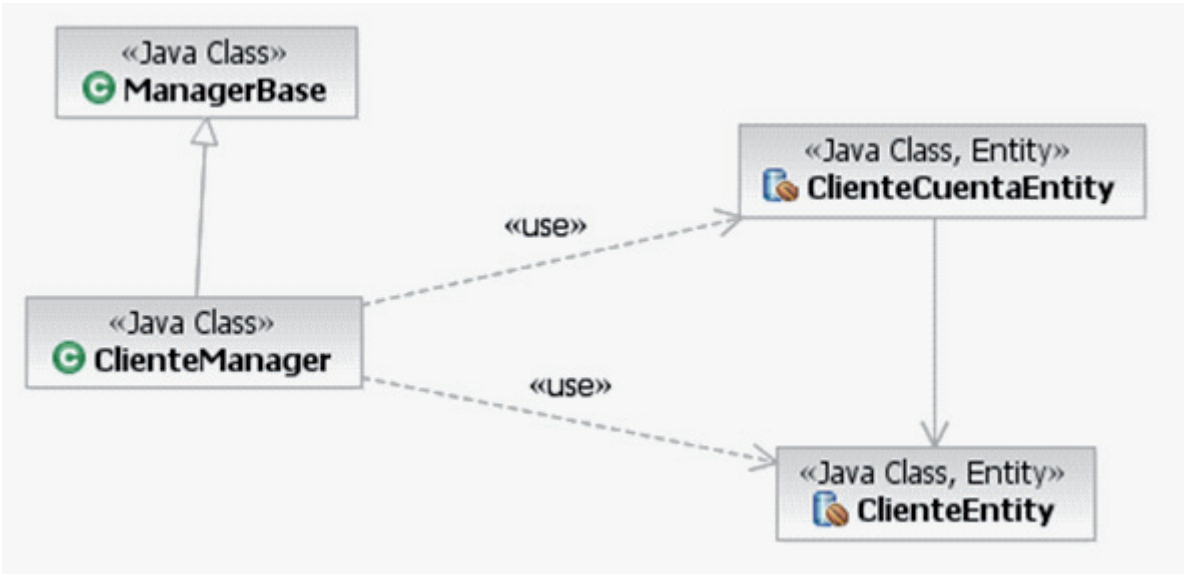


MANAGER Y ENTIDAD DE CLIENTE

Descripción del componente ClienteManager	
Nombre	**ClienteManager**
Versión	1.0
Tipo	Manager
Entidad que administra	ClienteEntity, ClienteCuentaEntity
Descripción	Este componente contendrá la lógica de negocio necesaria para administrar todo lo relacionado a un Cliente.
Métodos de negocio a implementar	
ObtenerClientesPorUsuario	Obtiene una lista de clientes pertenecientes a un Usuario
ObtenerCliente	Obtiene un cliente en base a su PK
ObtenerCuentas	Obtiene las cuentas asociadas de un cliente.
CrearCliente	Crea un cliente y lo persiste
ActualizarCliente	Actualiza los datos y cuentas de un cliente.
EliminarCliente	Asigna al cliente una fecha de baja. También asigna fecha de baja a todos los registros relacionados (ClienteCuentaEntity y CuentaEntity).

Descripción del componente ClienteEntity	
Nombre	**ClienteEntity**
Versión	1.0
Tipo	Entidad
PK	Long ColClienteId
Comentarios	Esta entidad se crea de forma automática con el Wizard JPA
Modificaciones	Se debe agregar el descriptor para indicar la secuencia a utilizar para el campo ColClienteId de la PK.
Atributos	
Los expuestos en el modelo de datos	
Métodos	
Se deberán implementar métodos getters y setters para todos los campos y entidades de la composición	

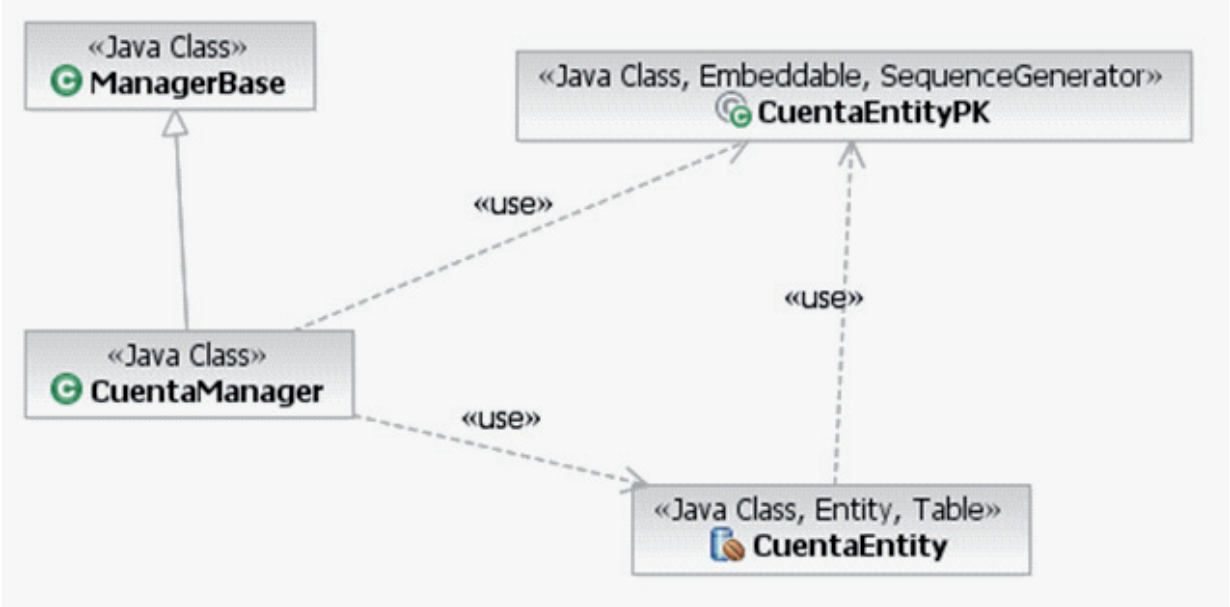
Descripción del componente ClienteCuentaEntity	
Nombre	**ClienteCuentaEntity**
Versión	1.0
Tipo	Entidad
PK	Long ColClienteCuentaId
Comentarios	Esta entidad se crea de forma automática con el Wizard JPA
Modificaciones	Se debe agregar el descriptor para indicar la secuencia a utilizar para el campo ColClienteCuentaId de la PK.
Atributos	
Los expuestos en el modelo de datos	
Métodos	
Se deberán implementar métodos getters y setters para todos los campos y entidades de la composición	



MANAGER Y ENTIDAD DE CUENTA

Descripción del componente CuentaManager	
Nombre	**CuenaManager**
Versión	1.0
Tipo	Manager
Entidad que administra	CuentaEntity
Descripción	Este componente contendrá la lógica de negocio necesaria para administrar una entidad Cuenta.
Métodos de negocio a implementar	
ObtenerCuenta	Obtiene un objeto Cuenta en base a la clave primaria
ObtenerCuentaPorCliente	Obtiene una lista de cuentas en base a un Cliente
CrearCuenta	Se encarga de crear una cuenta
EliminarCuenta	Asigna a una cuenta fecha de baja
ActualizarCuenta	Persiste en la base de datos los cambios realizados en una cuenta
ObtenerTodas	Obtiene una lista con todas las cuentas habilitadas

Descripción del componente CuentaEntity	
Nombre	**CuentaEntity**
Versión	1.0
Tipo	Entidad
PK	Objeto CuentaEntityPK
Comentarios	Esta entidad junto con su PK se crea de forma automática con el Wizard JPA
Modificaciones	Se debe agregar el descriptor para indicar la secuencia a utilizar para el campo colcuentaCuentaid de la PK.
Atributos	
Los expuestos en el modelo de datos	
Métodos	
Se deberán implementar métodos getters y setters para todos los campos y entidades de la composición	



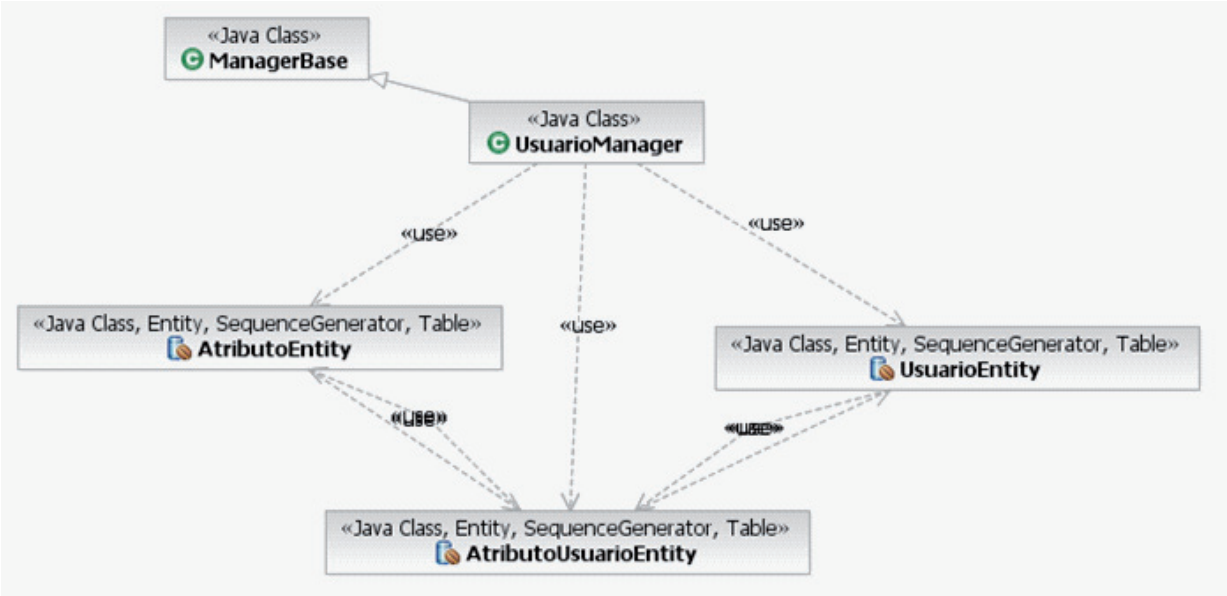
MANAGER Y ENTIDADES DE USUARIO

Descripción del componente UsuarioManager	
Nombre	**UsuarioManager**
Versión	1.0
Tipo	Manager
Entidad que administra	UsuarioEntity, AtributoUsuarioEntity, AtributoEntity
Descripción	Este componente contendrá la lógica de negocio necesaria para administrar una entidad Usuario con sus atributos. Los atributos que puede adquirir un usuario son los que se encuentran especificados en las entidades Atributo. Estos atributos se deberán configurar directamente en la tabla de la base de datos.
Métodos de negocio a implementar	
obtenerAtributosValoresUsuario	Obtiene todos los atributos de un usuario
obtenerValorAtributoUsuario	Obtiene el valor de un atributo de un usuario
ObtenerUsuario	Obtiene un objeto Usuario en base a un id de usuario
CrearUsuario	Se encarga de crear un Usuario con sus atributos
EliminarUsuario	Asigna a un usuario fecha de baja
ActualizarUsuario	Persiste en la base de datos los cambios realizados en un Usuario o en sus atributos

Descripción del componente UsuarioEntity	
Nombre	**UsuarioEntity**
Versión	1.0
Tipo	Entidad
PK	Long colusuarioId
Comentarios	Los objetos de esta clase contendrán los datos e información de un Usuario. La entidad estará compuesta por la entidad AtributoUsuarioEntity por lo cual la clase UsuarioEntity deberá implementar el Patrón de diseño Composite Entity. Esta entidad se crea de forma automática con el Wizard JPA
Modificaciones	Se debe agregar el descriptor para indicar la secuencia a utilizar para crear la PK.
Atributos	
Los expuestos en el modelo de datos	
Métodos	
Se deberán implementar métodos getters y setters para todos los campos y entidades de la composición	

Descripción del componente AtributoUsuarioEntity	
Nombre	**AtributoUsuarioEntity**
Versión	1.0
Tipo	Entidad
PK	Long colatributousuarioId
Comentarios	Los objetos de esta clase contendrán atributos de un Usuario. La entidad estará compuesta por la entidad AtributoEntity por lo cual la clase AtributoUsuarioEntity deberá implementar el Patrón de diseño Composite Entity. Esta entidad se crea de forma automática con el Wizard JPA
Modificaciones	Se debe agregar el descriptor para indicar la secuencia a utilizar para crear la PK.
Atributos	
Los expuestos en el modelo de datos	
Métodos	
Se deberán implementar métodos getters y setters para todos los campos y entidades de la composición	

Descripción del componente AtributoEntity	
Nombre	**AtributoEntity**
Versión	1.0
Tipo	Entidad
PK	colatributoId
Comentarios	Esta entidad se crea de forma automática con el Wizard JPA.
Atributos	
Los expuestos en el modelo de datos	
Métodos	
Se deberán implementar métodos getters y setters para todos los campos y entidades de la composición	



MANAGER Y BEANS DE DATOSFACTURA

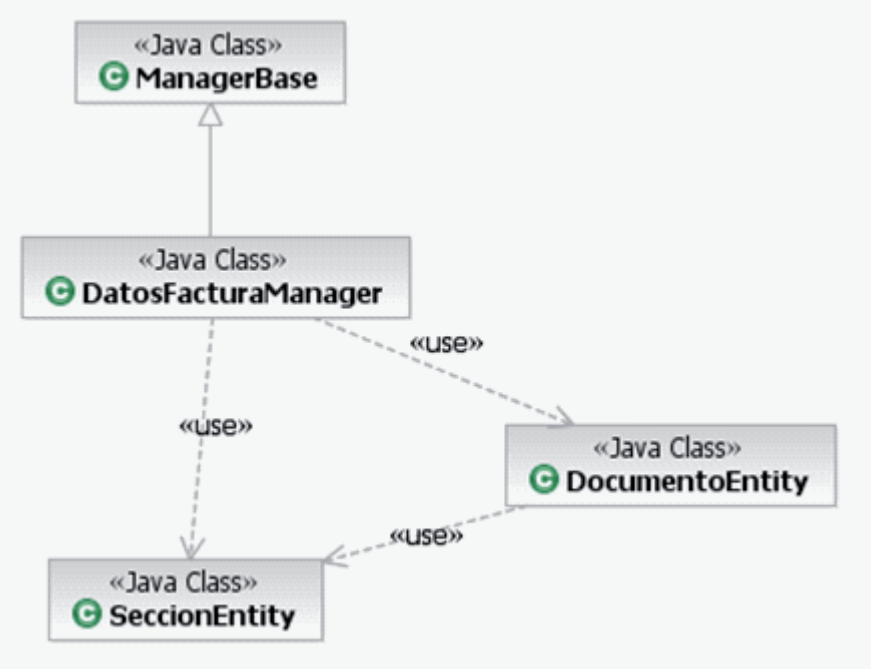
Descripción del componente DatosFacturaManager	
Nombre	**DatosFacturaManager**
Versión	1.0
Tipo	Manager
Bean que retorna	DocumentoEntity, SeccionEntity
Descripción	Este componente se encarga de realizar las llamadas a servicios web para: ● Obtener datos de las últimas 6 facturas de una cuenta ● Obtener el estado de pago de una factura. El estado incluye la siguiente información: o Secciones de la Factura o Link a Sección de la Factura o Estado de Pago o Disponibilidad y datos del comprobante ● Obtener el PDF de una factura que se encuentre en e2-vault.
Métodos de negocio a implementar	
obtenerDatosFacturas	Realiza los siguientes llamados a servicios web para obtener los datos de las facturas de una cuenta determinada: ● Datos de Facturas ● Datos de Pago
obtenerFactura	Realiza un llamado a un Servicio Web para obtener el PDF de una factura almacenada en e2-Vault

Descripción del componente DocumentoEntity	
Nombre	**DocumentoEntity**
Versión	1.0
Tipo	Bean
Comentarios	Este Bean contendrá la información de una Factura devuelta por el servicio de Datos de Factura y de Datos de Pago. A su vez este Bean tendrá una lista de secciones.

Atributos		
Atributo	**Tipo**	**Descripción**
Nro_Factura	String	Número de Factura
Fecha	Java.util.date	Fecha de vencimiento de la factura
Importe	Java.lang.double	Importe de la factura
CPE	String	Código de Pago Electrónico
Facturador	String	Opciones posibles: ATIS SIGECO
Origen	String	Opciones posibles: E2Vault FAOL
Pagada	Boolean	Retornará un “Si” si la factura fue saldada en su totalidad. En caso contrario retornará un “No”
Tiene Comprobante	Boolean	Retornará un “Si” si la factura tiene su comprobante de pago en ePagos. En caso contrario retornará un “No”.
Datos_Comprobante	String	Datos concatenados que se utilizarán para mostrar el comprobante.
Ciclo	String	Indica el ciclo de Facturación
Fecha de cierre	Java.util.date	Fecha utilizada en E2Vault como parte de clave primaria
Secciones	Collection	Colección de secciones de la factura que se llenará siempre y cuando el Origen del registro sea FAOL

Composición de la Colección "Secciones"		
Atributo	**Tipo**	**Descripción**
Sección	String	Identificador de la sección
Descripción de la sección	String	Descripción de la sección. Se mostrará en el combo de secciones.
Link	String	Link para obtener el archivo PDF con la sección correspondiente.
Operaciones		
Se deberán implementar métodos getters y setters para todos los campos.		

Descripción del componente SeccionEntity		
Nombre	**SeccionEntity**	
Versión	1.0	
Tipo	Entidad	
Comentarios	Este Bean contendrá la información de una sección	
Atributos		
Atributo	Tipo	Descripción
CPE	String	Código de Pago Electrónico
Cliente	String	Cliente ATIS/SIGECO
Cuenta	String	Cuenta ATIS/SIGECO
NroFactura	String	Número de Factura
Facturador	String	Facturador
Ciclo	String	Indica el ciclo de Facturación
Fecha de cierre	Java.util.date	Fecha utilizada en E2Vault como parte de clave primaria
Stream	Byte[]	Stream de bytes que será el archivo PDF a visualizar.
Operaciones		
Se deberán implementar métodos getters y setters para todos los campos.		



A continuación se especifican los parámetros de entrada y salida propuestos para los servicios.

Servicio para obtener datos de facturas:

Nombre	Descripción	Tipo
Cliente	Cliente ATIS/SIGECO	Entrada
Cuenta	Cuenta ATIS/SIGECO	Entrada
Resultado	Colección de datos de factura	Salida
Codigo_Error	Código de error devuelto por E2 Vault	Salida

Resultado

Nombre	Descripción
Nro_Factura	Número de Factura
Fecha	Fecha de vencimiento de la factura
Importe	Importe de la factura
CPE	Código de Pago Electrónico
Facturador	Opciones posibles: • ATIS • SIGECO
Origen	Opciones posibles: • E2Vault • FAOL
Ciclo	Indica el ciclo de Facturación
Fecha de cierre	Fecha utilizada en E2Vault como parte de clave primaria
Secciones	Colección de secciones de la factura que se llenará siempre y cuando el Origen del registro sea FAOL

Sección

Nombre	Descripción
Sección	Identificador de la sección
Descripción de la sección	Descripción de la sección. Se mostrará en el combo de secciones.
Link	Link para obtener el archivo PDF con la sección correspondiente.

Servicio para obtener datos de pago de la factura:

Nombre	Descripción	Tipo
CPE	Código de Pago Electrónico	Entrada
NRO_FACTURA	Número de Factura	Entrada
CLIENTE	Cliente ATIS/SIGECO	Entrada
CUENTA	Cuenta ATIS/SIGECO	Entrada
PAGADO	Indica si la factura fue paga o no	Salida
DATOS_COMPROBANTE	Retorna datos para reconstruir el comprobante de pago.	Salida

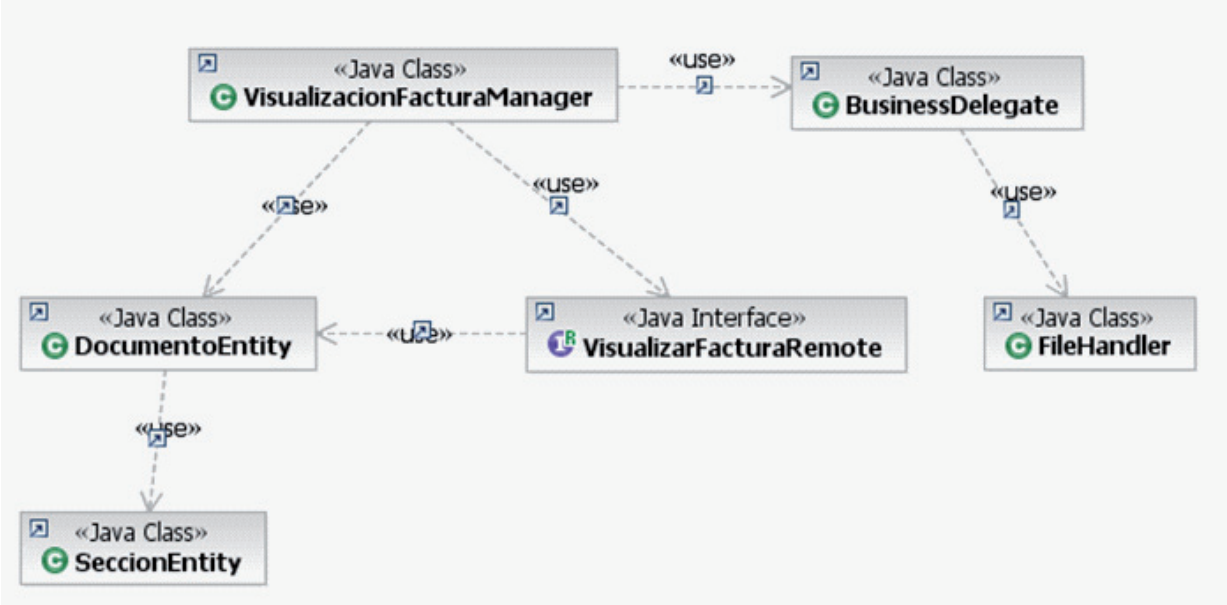
Servicio para obtener un PDF de e2-Vault

Nombre	Descripción	Tipo
CPE	Código de Pago Electrónico	Entrada
Cliente	Cliente ATIS/SIGECO	Entrada
Cuenta	Cuenta ATIS/SIGECO	Entrada
Ciclo	Indica el ciclo de Facturación	Entrada
Fecha de cierre	Fecha utilizada en E2Vault como parte de clave primaria	Entrada
Nro de Factura	Nro Fc ATIS/SIGECO	Entrada
Stream	Stream de bytes que será el archivo PDF a visualizar.	Salida
Codigo_Error	Código de error devuelto por E2 Vault	Salida

DESCRIPCIÓN DE VISUALIZACIONFACTURA (FACADE)

Definición de Elemento Sw VisualizacionFacturaFacade	
Nombre	**VisualizacionFacturaFacade**
Versión	1.0
Tipo	Clase
Patrón de Diseño	Facade
Descripción	Esta clase contendrá la implementación de los métodos de negocios a utilizar desde la capa de presentación
Interface Remota	VisualizacionFacturaFacade
Métodos de negocio a implementar	
obtenerClientes	Obtiene todos los clientes que tiene asociado un usuario.
obtenerCuentas	Obtiene todas las cuentas de un Cliente. (Llama al ClienteManager)
obtenerCuenta	Obtiene una cuenta determinada en base a una PK (Llama al CuentaManager)
obtenerValorAtributoUsuario	Obtiene el valor de un atributo de un usuario (Llama al UsuarioManager)
obtenerDatosFactura	Obtiene los datos de las facturas de una cuenta (Llama al DocumentoFacturaManager)
obtenerFactura	Obtiene el PDF de una factura que se encuentra en E2-Vault (Llama al DocumentoFacturaManager)

Componente COM-DT001-002 (Presentación)



CLASE BUSINESSDELEGATE

Definición de Elemento Sw BusinessDelegate	
Nombre	BusinessDelegate
Versión	1.0
Tipo	Clase
Patrón de Diseño	BusinessDelegate
Descripción/ Pseudocódigo	Se encargar de las siguientes tareas: ● Determinar el Facade o WebService a utilizar ● Establecer un contexto en el Container EJB ● Instanciar remotamente el facade

CLASE BUSINESSDELEGATE

Definición de Elemento Sw FileHandler	
Nombre	BusinessDelegate
Versión	1.0
Tipo	Clase
Descripción/ Pseudocódigo	Se encargara de leer los datos necesarios para establecer una conexión al Container EJB desde un archivo de configuración.

COMPONENTE VISUALIZACIONFACTURA_STUB

Definición de Elemento Sw VisualizacionFactura_STUB	
Nombre	VisualizacionFactura_STUB
Versión	1.0
Tipo	Componente
Descripción/ Pseudocódigo	Es un componente JAR que contendrá las clases necesarias para poder instanciar el Facade de forma remota. Este Jar se deberá crear con un script provisto por el RAD 7.5

CREACIÓN DE UN STUB DE UN EJB CON RAD 7.5

1. Seleccionar el proyecto EJB y exportar el mismo. Seleccionar Java ➔ Archivo JAR
2. Seleccionar el destino C:\[NombreEJB].jar y hacer clic en finalizar
3. Abrir una línea de comando y posicionarse en el directorio <RAD-HOME>/runtimes/base_v61/bin
4. Ejecutar el siguiente comando para crear el Stub:

createejbstubs c:\[NombreEJB].jar –newfile

5. La librería resultante se llamará [NombreEJB]_withStubs.jar. La misma deberá utilizarse en el proyecto Faces para poder instanciar de forma remota un Facade

CLASE VISUALIZACIONFACTURAMANAGER

Definición de Elemento Sw VisualizacionFacturaManager

Nombre	VisualizacionFacturaManager
Versión	1.0
Tipo	ManageBean
Descripción/ Pseudocódigo	Es la clase que se encargará de manejar la página JavaFaces. ● nessDelegate ● Invocará a los procesos de Negocio del Facade para mostrar los datos de la factura de una cuenta seleccionada
Métodos de negocio a implementar	
getMediosPago	Obtiene una lista de SelectItems con los medios de pago habilitados
getDocumentos	Obtiene los datos de las facturas de la cuenta seleccionada
getUnidadNegocioCliente	Obtendrá la unidad de Negocio del Cliente logueado
getCuentas	Generará una lista de SelectItems con las cuentas del Cliente seleccionado. Un ítem estará compuesto por el ANI y por el titular de la Cuenta.