ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

FACULTAD DE INFORMATICA Y ELECTRONICA

ESCUELA DE INGENIERIA EN SISTEMAS

"ESTUDIO COMPARATIVO DE SERVIDORES DE APLICACIONES PARA DESARROLLO DE SOFTWARE CON SOA SOBRE PLATAFORMAS JAVAEE CASO PRACTICO TRANSPORTES PATRIA"

TESIS DE GRADO

PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS

WILSON JAVIER ROMERO GUILLEN

ANIBAL GIOVANY HERRERA MONCAYO

RIOBAMBA ECUADOR

2011

AGRADECIMIENTO

Como una muestra de nuestro cariño y agradecimiento, por todo el apoyo brindado a las personas que incondicionalmente tendieron su mano, ya que gracias a su apoyo y consejo nos encaminaron e impulsaron para alcanzar esta meta. La cual constituye la herencia más valiosa que podemos recibir, a la Escuela Superior Politécnica de Chimborazo, a la Escuela de Ingeniería en Sistemas, a nuestros padres, hermanos, docentes y amigos.

DEDICATORIA

La fe, el esfuerzo y optimismo dedicado a lo largo de los años de estudio, son el fruto de la gente que creyó en nosotros, apoyándonos en todo sentido extendiendo la mano a través de la educación. Por ello este trabajo está dedicado a las personas que a lo largo de nuestras vidas han dado la formación de ser persona.

Con mucho cariño a nuestros padres, quienes nos brindaron su apoyo moral y económico en cada etapa de nuestra carrera.

NOMBRE	FIRMA	FECHA
Ing. Iván Menes		
DECANO	**…..**	**.....**
Ing. Raúl Rosero		
DIRECTOR DE LA		
ESCUELA	**…..**	**.....**
Ing. Raúl Rosero		
DIRECTOR DE TESIS	**…..**	**.....**
Ing. Landy Ruíz		
MIEMBRO DEL TRIBUNAL	**.....**	**.....**
Tlgo. Carlos Rodríguez		
DIRECTOR DPTO	**.....**	**.....**
DOCUMENTACIÓN		
NOTA DE LA TESIS	**.....**	

RESPONSABILIDAD DEL AUTOR

Nosotros, Wilson Javier Romero Guillen y Anibal Giovany Herrera Moncayo, somos responsables de las ideas y doctrinas y resultados expuestos en esta Tesis y el patrimonio de la misma pertenece a la Escuela Superior Politécnica de Chimborazo

_____ _____

Wilson Javier Romero Guillen **Anibal Giovany Herrera Moncayo**

ABREVIATURAS

AOP	Aspect Oriented Programming
BI	Business intelligence
CRM's	Sistema de Gestión de Clientes
DNS	Domain Name Server
EAR	Enterprise Archive, Archivo Empresarial
EJB	Enterprise JavaBeans
ERP´s,	Planificación de Recursos Empresariales
ESB	Enterprise service bus
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JAR	Java Archive, Archivo Java
JDBC	Java Data Base Connectivity
JMX	Java Management Extensions
JNDI	ava Naming and Directory Interface
JSP	JavaServer Pages
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
QoS	Quality of service (Calidad de Servicio)
SA	Servidor de Aplicaciones
SOA	Arquitectura Orienta a Servicios
SOAP	Simple Object Access Protocol.
SQLJ	SQL en programas de Lenguaje de programación Java
SSL	Secure Socket Layer
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator

XML	Extensible Markup Language
WAS	WebSphere Aplication Server
WAR	Web Aplication
WSDL	Lenguaje de Descripción de Servicios Web

•

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPITULO I

MARCO REFERENCIAL

- **ANTECEDENTES**

PLANTEAMIENTO DEL PROBLEMA.

- **DESCRIPCIÓN DEL OBJETO DE ESTUDIO.**

Un servidor de aplicaciones es un producto basado en un componente que se encuentra en el plano medio de la arquitectura central de un servidor. Proporciona servicios de 'middleware', es decir, trabaja como un intermediario para la seguridad y el mantenimiento, además de proveer acceso a los datos.

Un ejemplo común del uso de servidores de aplicación (y de sus componentes) son los sitios de internet que presentan una gran variedad de servicios, que permiten a las empresas tales como TRANSPORTES PATRIA la gestión y divulgación de su información, y un punto único de entrada a los usuarios internos y externos. Teniendo como base un servidor de aplicaciones, dichos portales permiten tener acceso a información y servicios (como servicios Web) de manera segura y transparente desde cualquier dispositivo. El desarrollo de soluciones que permiten este tipo de interacción

y funcionalidad se lo hace siguiendo los lineamientos de la arquitectura orientada a servicios.

- **SITUACIÓN ACTUAL DEL OBJETO DE ESTUDIO.**

En la actualidad los servidores de aplicación incorporan capacidades que permiten la elaboración de soluciones empresariales como clustering y administración avanzada. Además ofrecen funcionalidades imprescindibles para implantaciones escalables y de alta fidelidad, permitiendo en este caso a las empresas la incorporación de nuevos módulos a sus sistemas sin que esto les represente pérdidas de tiempo y elevados costos.

Actualmente el manejo de la información financiera dentro de TRANSPORTES PATRIA, se lleva a cabo mediante el soporte de dos aplicaciones informáticas, que permiten realizar el registro de las transacciones, pero que no trabajan de forma sincronizada sobre la misma base de datos. Además las once agencias ubicadas en todo el país envían este tipo de información hasta la matriz a través de diferentes medios de almacenamiento y se lo hace únicamente a fin de mes. Esta es la razón por la que se requiere la implementación de un sistema con SOA para que permita la actualización permanente de la información desde cualquier punto del país.

Por esto es importante la evaluación de diferentes alternativas previo a la elección de un servidor de aplicaciones para la futura implantación de un sistema con estas características. Este proceso de análisis debe estar enfocado a los distintos parámetros recomendados por expertos y empresas proveedoras de servidores, siempre buscando que el conjunto de características de un producto como tal se ajusten a las capacidades ideales que la empresa requiere. Entre los parámetros que se toman en cuenta, actualmente, antes de la elección de un servidor de aplicaciones para una determinada solución, se plantean los siguientes:

- Rapidez y rendimiento antes, durante y después del despliegue de una aplicación.
- Capacidad de Clustering, es decir la posibilidad de agrupar servidores para lograr mayor escalabilidad.
- Tareas de Administración Centralizada, a través de consolas o terminales que permitan gestionar: los clústeres, despliegues de aplicaciones, la carga de transacciones, variables de sesión, entornos de configuración y ejecución, etc.
- Interoperabilidad entre distintas arquitecturas y plataformas.
- Deben permitir el desarrollo de recursos de apoyo empresarial.
- Proporcionar un enfoque centralizado para suministrar SOA utilizando servicios Web.
- Integración con Entornos de Desarrollo Integrado (IDE) – lo que permite a los desarrolladores desplegar aplicaciones SOA diseñando workflows de BPEL (Business Process Execution Languaje).

Si bien el término es aplicable a todas las plataformas de software, hoy en día el término servidor de aplicaciones se ha convertido en sinónimo de la plataforma Java EE (antes J2EE) de Sun Microsystems.

Como consecuencia del éxito del lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a un servidor de aplicaciones Java EE.

WebSphere (IBM) y WebLogic (Oracle, antes BEA Systems) están entre los servidores de aplicación Java EE privativos más conocidos. EAServer (Sybase Inc.) es también conocido por ofrecer soporte a otros lenguajes diferentes a Java, como PowerBuilder. El servidor de aplicaciones JOnAS, desarrollado por el consorcio ObjectWeb, fue el primer servidor de aplicaciones libre en lograr certificación oficial de compatibilidad con

J2EE. JBoss es otro servidor de aplicaciones libre y muy popular en la actualidad, así como el GlassFish de SUN.

Java EE provee estándares que permiten a un servidor de aplicaciones servir como "contenedor" de los componentes que conforman dichas aplicaciones. Estos componentes, escritos en lenguaje Java, usualmente se conocen como Servlets, Java Server Pages (JSPs) y Enterprise JavaBeans (EJBs) y permiten implementar diferentes capas de la aplicación, como la interfaz de usuario, la lógica de negocio, la gestión de sesiones de usuario o el acceso a bases de datos remotas.

La portabilidad de Java también ha permitido que los servidores de aplicación Java EE se encuentren disponibles sobre una gran variedad de plataformas, como Unix, Microsoft Windows y GNU/Linux.

- **ANÁLISIS DEL OBJETO DE ESTUDIO.**

Aplicaciones Corporativas

Las aplicaciones Corporativas son el eje de las operaciones empresariales ya que regulan las actividades esenciales que ayudan a mejorar la satisfacción del cliente y a acelerar el crecimiento de la empresa. Los empresarios comparten información y se comunican con sus empleados, clientes, socios y proveedores de forma diaria. Utilizan un recurso empresarial para planificar sus soluciones y los distintos procesos de la cadena de proveedores. Para tomar decisiones empresariales de una forma más rápida y efectiva es imprescindible poder acceder a la información y así administrarla e interpretarla.

Arquitectura Orientada a Servicios

SOA es un marco de trabajo conceptual que permite a las organizaciones unir los objetivos de negocio con la infraestructura de TI integrando los datos y la lógica de negocio de sus sistemas separados.

SOA permite la creación de sistemas altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma estándar de exposición e invocación de servicios (comúnmente pero no exclusivamente servicios web), lo cual facilita la interacción entre diferentes sistemas, incluso heterogéneos, propios o de terceros.

SOA define las siguientes capas de software:

- Aplicaciones básicas Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- De exposición de funcionalidades Donde las funcionalidades de la capa aplicativas son expuestas en forma de servicios (servicios web).
- De integración de servicios Facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración;
- De composición de procesos Que define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio.
- De entrega donde los servicios son desplegados a los usuarios finales.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

Servidor de Aplicaciones

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http.

Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Un servidor de aplicaciones es un producto basado en un componente que se encuentra en el plano medio de la arquitectura central de un servidor. Proporciona servicios de 'middleware', es decir, trabaja como un intermediario para la seguridad y el mantenimiento, además de proveer acceso a los datos.

Un servidor de aplicación maneja la mayoría de las transacciones relacionadas con la lógica y el acceso a los datos de la aplicación (esto se solía llamar 'centralización', hace algún tiempo...). La ventaja principal de un servidor de aplicaciones es la facilidad para desarrollarlas, puesto que éstas no necesitan ser programadas y en cambio, se arman a partir de módulos provistos por el servidor de aplicaciones.

- **DELIMITACIÓN**

El presente estudio se encontrará enfocado en un análisis comparativo de los servidores de Aplicaciones, para determinar la mejor opción que permita la implementación del sistema de control financiero de la empresa TRANSPORTES PATRIA con Arquitectura Orientada a Servicios, de tal manera que se asegure la integración de las aplicaciones que con el futuro se vayan integrando a esta plataforma, en busca de ir representando y respaldando todos los procesos de negocio de la empresa.

TRANSPORTES PATRIA necesita acceder a los datos de las transacciones de las diferentes agencias y así efectuar su respectivo seguimiento y control contable, siempre buscando mejorar los tiempos de respuesta actuales y la optimización de los recursos en la ejecución de las tareas.

Entre los servidores de aplicación que se considerarán para el presente estudio tenemos:

- WebSphere Application Server de IMB

- Oracle WebLogic Application Server
- RedHat JBoss Application Server
- Sun GlassFish Enterprise Server (formalmente Sun Java System Application Server)

El sistema operativo servidor puede no tiene restricciones, ya que se puede tratar de cualquier entorno que soporte JavaEE, dentro de los cuales se dispone de:

- Linux, en cualquiera de sus distribuciones para servidor
- OpenSolaris

El motor de base de Datos elegido es MySQL, y la herramienta de desarrollo principal es NetBeans, que permite el trabajo con JavaEE y además brinda la posibilidad de la implementación de Servicios Web, necesarios para esta tipo de soluciones.

- **FORMULACIÓN DEL PROBLEMA**

¿Cuál es el mejor servidor de aplicaciones para el desarrollo de software que permita implementar Arquitectura Orientada a Servicios?

- **SISTEMATIZACIÓN**

- ¿Qué es SOA?
- ¿Qué es Servidor de Aplicaciones?
- ¿Cuáles son las ventajas y desventajas de los Servidores de Aplicaciones existentes actualmente?
- ¿Cuáles son las herramientas de actualidad en el medio para la implementación de Software con SOA?
- ¿Qué modelos y estándares son empleados para lograr la integración de los componentes de un Sistema con SOA en un Servidor de Aplicaciones?

- ¿Cómo se encuentra estructurada la Arquitectura básica de un Servidor de Aplicaciones?
- ¿Cuál es la especificación de SOA para la implementación de Soluciones Corporativas que permitan la integración de plataformas heterogéneas?

- **JUSTIFICACIÓN DEL PROYECTO DE TESIS.**

Justificación Teórica.

Hoy en día el desarrollo de software con SOA necesariamente está presente en la gran mayoría de soluciones empresariales. Es importante entonces el conocimiento de los diferentes parámetros que deben ser evaluados antes de la elección de un servidor de Aplicaciones para la implementación de software con SOA. Las principales ventajas de la tecnología de los servidores de aplicación es la centralización y la disminución de la complejidad del desarrollo de aplicaciones, dado que estas no necesitan ser programadas desde cero; en su lugar, estas son ensambladas desde bloques provistos por el servidor de aplicación.

Justificación Práctica.

Las transacciones de TRANSPORTES PATRIA necesitan ser registradas de forma adecuada para que se permita el tratamiento de la información de forma fácil, y evitando cualquier tipo de inconsistencia. Además la información financiera de TRANSPORTES PATRIA que se genera desde las distintas agencias ubicadas en el país por concepto de ventas de pasajes y encomiendas necesita de la misma forma ser actualizada diariamente hacia la matriz Riobamba. Se plantea entonces, la elaboración de una solución informática, que permita la alimentación de la información hacia la base de datos central de la empresa, y que represente los distintos procesos de negocio que se llevan a cabo.

Para ello se realizará un estudio que permita la elección del servidor de aplicaciones adecuado para el desarrollo de software con arquitectura orientada a servicios, necesario para la implementación de un sistema con las características mencionadas anteriormente.

- **OBJETIVOS**

2.1.1.8.1 OBJETIVO GENERAL

Realizar un Estudio Comparativo que facilite la elección de un Servidor de Aplicaciones sobre plataformas JavaEE, para el desarrollo de Software con Arquitectura Orientada a Servicios (SOA).

o 2.1.1.8.2 OBJETIVOS ESPECÍFICOS:

- Establecer las diferencias entre Tecnologías de Servidor y Servidor de Aplicaciones.
- Conocer las ventajas de las Aplicaciones que se encuentran implementadas sobre Arquitecturas Orientadas a Servicios.
- Establecer los componentes que integran un servidor de aplicaciones que soporte Arquitectura Orientada a Servicios.
- Comparar cuatro Servidores de Aplicaciones seleccionados, de los existentes en el medio, para la implementación de Software con SOA.
- Verificar los tiempos de respuesta, que se conseguirán luego de la selección del servidor adecuado para la implementación de SOA

- **HIPÓTESIS.**

La utilización de un servidor de aplicaciones adecuado, permitirá la interoperabilidad entre la Arquitectura Orientada a Servicios y la plataforma JavaEE, reduciendo los tiempos de respuesta de acceso a la información.

- **MÉTODOS Y TÉCNICAS**

El método utilizado como guía para la presente investigación es el método analítico, el cual contempla los siguientes puntos que involucran el desarrollo de esta tesis:

- El planteamiento del problema motivo del presente trabajo.
- El apoyo del proceso previo a la formulación de la Hipótesis.
- El proceso de recopilación de la información necesaria.
- Análisis e interpretación de Resultados.
- El proceso de Comprobación de la Hipótesis, etc.

También se utilizará como complemento del presente trabajo al método Analítico – Sintético, por cuanto, este establece el procedimiento necesario para la recopilación, análisis e integración de resultados necesarios para el planteamiento de la metodología propuesto por los autores, en base a la información recopilada.

CAPITULO II

ARQUITECTURA ORIENTADA A SERVICIOS

- **INTRODUCCIÓN SOA**

La arquitectura orientada a servicios (SOA) es un tema importante en la informática empresarial especialmente una SOA basada en WEB SERVICES en el sentido de acelerar drásticamente el proceso de desarrollo de aplicaciones.

También se la aprecia como una manera de construir aplicaciones y sistemas que sean más adaptables, y de ese modo, y de allí que la TI vaya volviéndose más ágil en dar respuestas a las necesidades cambiantes del negocio.

SOA es una evolución de la llamada computación distribuida**,** basada en el paradigma de pregunta/respuesta para aplicaciones síncronas y asíncronas. En ellas la lógicas de negocio o las funciones individuales son presentadas como servicios para aplicaciones consumidoras /clientes. La clave de estos servicios es su naturaleza desacoplada, la interfaz de servicios es indiferente de la implementación de los mismos. Por ejemplo, un servicio puede ser implementado tanto en .Net como en JavaEE, y la aplicación que consume el servicio incluso puede estar en una plataforma y lenguaje diferente a esto dos.

Todas las arquitecturas SOA tienen las siguientes características clave:

Los servicios SOA auto describen su interfaz en documentos XML que son independientes de la plataforma. WSDL (Web Service Description Language) es el lenguaje utilizado para describir los servicios.

Los servicios SOA se comunican con mensajes formalmente definidos vía un esquema XML. La comunicación entre consumidores y proveedores de servicios típicamente ocurre en un ambiente heterogéneo, con poco o ningún conocimiento sobre el proveedor.

Los servicios SOA son mantenidos en una empresa por un registro que actúa como un directorio. Las aplicaciones pueden mirar los servicios en este registro e invocarlos. UDDI (Universal Description Definition and Integration).

Cada servicio SOA tiene una calidad de servicio asociada con el (QoS).Algunos de los ejemplos clave de la QoS son los requerimientos de seguridad, como la autenticación y la autorización, mensajería confiable y políticas definiendo quién puede acceder a los servicios.

- **DEFINICIONES**

SOA (Arquitectura Orientada a Servicios). Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir. (W3C).

SOA (Arquitectura Orientada a Servicios). Estilo resultante de políticas, prácticas y frameworks que permiten que la funcionalidad de una aplicación se pueda proveer y consumir como conjuntos de servicios, con una granularidad relevante para el consumidor. Según CBDI

SOA (Arquitectura Orientada a Servicios). Es un estilo de arquitectura que promueve descomponer la lógica funcional de una aplicación en unidades autónomas denominadas servicios.

Figura II.1 SOA

SOA (Arquitectura Orientada a Servicios). Es un conjunto de Servicios operando conjuntamente con algún fin específico, el mismo que permitirá, bajo acoplamiento, granularidad gruesa, componentes, mensajería, etc.

SOA (Arquitectura Orientada A Servicios). Es un enfoque para diseñar y construir soluciones de negocio a partir de componentes independientes que exponen funciones como servicios accesibles por otros componentes a través de interfaces

o **CARACTERÍSTICAS**

El SOA tiene estas características:

- ✓ Servicios descubribles
- ✓ Servicios modulares
- ✓ Servicios interoperables
- ✓ Servicios de acoplamiento flojo
- ✓ Servicios con interfaces de granularidad gruesa
- ✓ Servicios transparentes de localización
- ✓ Servicios mediante composición
- ✓ Servicios autorecuperables

Servicios descubribles

Un consumidor de servicios que necesita un servicio, descubre que servicio usar basado en un conjunto de criterios en tiempo de ejecución. El consumidor de servicios le pregunta a un registro por un servicio que llene sus necesidades. Los consumidores no necesitan ninguna información sobre el servicio en tiempo de compilación.

Las interfaces de los servicios son descubiertas en tiempo de ejecución y los mensajes son construidos dinámicamente. El consumidor del servicio no conoce el formato de los mensajes de solicitud o respuesta o la localización del servicio hasta que es realmente necesario.

Servicios modulares

Uno de los aspectos más importantes de las SOAs es la modularidad. Un servicio soporta un conjunto de interfaces, las cuales deben ser cohesivas. Esto significa que todas las interfaces deben relacionar entre sí en el contexto de un módulo. La modularidad debe ser inherente al diseño de servicios que soportan una aplicación. De esta forma los servicios pueden ser fácilmente agregados con unas pocas dependencias bien conocidas.

Existen cinco criterios para determinar si un componente es suficientemente modular:

- ✓ Descomposición modular
- ✓ Composición modular
- ✓ Entendimiento modular
- ✓ Continuidad modular
- ✓ Protección modular

Descomposición modular

La descomposición modular de un servicio se refiere a la separación de una aplicación en muchos módulos pequeños. Cada módulo es responsable por una función única y distinta en la aplicación. Esto aveces se le conoce como ‘diseño de arriba hacia abajo‘. La principal meta de la descomposición modular es la reutilización.

Composición modular

La composición modular de un servicio se refiere a la producción de servicios de software que pueden ser combinados como un todo con otros servicios para producir nuevos sistemas. Los diseñadores de servicios deben crear servicios suficientemente independientes para reusarse en aplicaciones completamente diferentes de las que fueron originalmente concebidos.

Entendimiento modular

El entendimiento modular de un servicio es la habilidad de una persona para entender la función de un servicio sin tener ningún conocimiento de otros servicios. Este es especialmente importante para servicios, debido a que cualquier cliente desconocido puede encontrar y usar un servicio en cualquier momento.

Continuidad modular

La continuidad modular de un servicio se refiere al impacto que el cambio de un servicio provoca en otros servicios o en los consumidores del servicio. Una interfaz que no oculta suficientemente los detalles de la implementación de un servicio crea un efecto dominó cuando algún cambio es necesario.

Protección modular

La protección modular de un servicio es suficiente si un condición anormal en el servicio no provoca una cascada a otros servicios o consumidores. Los fallos en la operación de un servicio no deben impactar la operación de un cliente u otro servicio o el estado de sus datos internos.

Interoperabilidad

Interoperabilidad consiste en la habilidad de los sistemas para utilizar diferentes plataformas y lenguajes para comunicarse entre ellos. Cada servicio provee una interfaz que puede ser invocada mediante un tipo de conector. Un tipo de conector interoperable consiste de un protocolo y un formato de datos para cada uno de los clientes potenciales del servicio.

Acoplamiento flojo

El acoplamiento se refiere al número de dependencias entre módulos. Existen dos tipos de acoplamiento: flojo y ajustado. Los módulos de acoplamiento flojo tienen unas pocas dependencias bien conocidas. Los módulos de acoplamiento ajustado tienen muchas dependencias desconocidas. Las SOAs proponen el acomplamiento flojo entre consumidores y proveedores de servicio.

Granularidad de las interfaces

El concepto de granularidad aplica a servicios en dos formas:

- ✓ Esta aplica al alcance del dominio que el servicio completo implementa.
- ✓ Esta también aplica al alcance del dominio que cada método implementa dentro de la interfaz.

Estos niveles de granularidad están relacionados entre sí. Las SOAs proponen la granularidad gruesa de interfaces

Transparencia de localización

La transparencia de localización es una característica clave de las SOAs. Los consumidores de un servicio no conocen la localización del servicio hasta que ellos lo buscan en el registro. El enlazado dinámico al servicio en tiempo de ejecución permite que la implementación del servicio se mueva de una localización a otra sin el conocimiento de los clientes.

Composición del servicio

La composición de un servicio está relacionada a su estructura modular. La estructura modular posibilita que los servicios sean ensamblados en aplicaciones de las cuales el desarrollador no tenía noción cuando diseñó el servicio. El uso de servicios evaluados y preexistentes mejora la calidad de un sistema.

Existen tres formas de composición:

- ✓ Aplicación: Se compone de servicios, componentes, y la lógica de la aplicación que enlaza estas funciones juntas para un propósito específico.

- ✓ Federación de servicios: Son colecciones de servicios administrados en forma conjunta en un gran dominio de servicio.
- ✓ Orquestación de servicios: Es la ejecución de una transacción única que impacta uno o más servicios en una organización.

Auto recuperación

Con el tamaño y la complejidad de las aplicaciones distribuidas modernas, cada vez es más importante la habilidad de un sistema de recobrarse de los errores. Un sistema auto recuperable es uno que tiene la habilidad de recuperarse de errores sin intervención humana durante la ejecución.

SOA en el nivel de implementación:

1. SOA separa en niveles el proceso, por ejemplo la interfaz de servicio y el negocio; el negocio y el acceso a datos.
2. El acceso a servicios es transparente en la red, es decir a los consumidores de servicios (Aplicaciones front end) no les interesa conocer la forma en que estos responden a sus solicitudes.
3. Los servicios deben ser en la medida de lo posible especializados en una tarea; eso con la finalidad de incrementar el grado de reutilización de los componentes.
4. Los servicios están dispersos por toda la red (Repositorio de servicios)
5. Los consumidores invocan los servicios sin conocer donde se ubican y otros servicios se encargan de direccionarlos (Service Bus)

o **VENTAJAS Y DESVENTAJAS**

Ventajas

1. **La arquitectura SOA ayuda a mejorar la agilidad y flexibilidad de las organizaciones.**

   Las empresas deben ser capaces de crear y producir nuevos productos y servicios para unos clientes y ciudadanos que son cada vez más exigentes. El aumento de

la colaboración con los clientes y proveedores, y la mayor capacidad para interpretar los datos de los clientes, proporcionan a las organizaciones los medios necesarios para interpretar los cambios del mercado de una forma más precisa y rápida. Lo que necesitan en estos momentos es conseguir que sus procesos de negocio sean capaces de ser adaptados al menos al mismo ritmo. Este dinamismo exige un nuevo conjunto de capacidades tecnológicas que permitan adaptar rápidamente los sistemas informáticos.

El pensamiento tecnológico tradicional, que normalmente intentaba crear una aplicación nueva para cada proceso nuevo, nunca ha sido capaz de generar tal agilidad. Las aplicaciones se desarrollaban normalmente en momentos diferentes, con diferentes intenciones, plataformas, conjuntos de usuarios y niveles de servicio, y suponían diferentes ciclos de mantenimiento, mejoras y presupuestos. Haciendo un análisis retrospectivo, no nos sorprende que los esfuerzos por integrar las aplicaciones y los sistemas de una organización pudieran ser tan laboriosos y costosos de implantar y mantener.

Por el contrario, la arquitectura SOA se centra en las capacidades, no en las aplicaciones. SOA contempla la arquitectura de toda la empresa, incluidos los procesos de negocio y las tecnologías de la información. Además, el alto nivel de desacoplamiento e interoperabilidad proporcionado por la arquitectura SOA permite un alto grado de reutilización (interno y externo) y de parametrización. Todo ello redunda en una mayor facilidad y flexibilidad para adaptar y mejorar los procesos de las organizaciones según los cambios de prioridad del negocio.

2. **La arquitectura SOA permite una “personalización masiva” de las tecnologías de la información.**

   La personalización masiva es un concepto que se ha tomado prestado de los procesos de fabricación, donde al combinar de distinta manera los módulos estándar, se puede dar forma a un producto individualizado dentro de la

infraestructura masiva de producción. Mediante la arquitectura SOA se puede aplicar el mismo principio a la tecnología de una organización y, como consecuencia, a los procesos de negocio habilitados por dicha tecnología. Así por ejemplo, en una gran compañía de telecomunicaciones, la arquitectura SOA ha permitido acelerar el proceso de creación e integración de nuevos servicios, y abaratar sus costes, lo que ha permitido desarrollar complejas políticas de precios y contratación mejor adaptadas a segmentos específicos de clientes.

Frecuentemente, la información necesaria para desarrollar nuevos servicios o productos ya existe dentro de los sistemas de la organización, y la arquitectura SOA crea una forma más fácil y rápida (y más barata) de acceder a ella que en el pasado, y de utilizarla en la unidad que gestiona los clientes. Ésta es, en efecto, la capacidad de personalizar productos y servicios a gran escala, mientras se utiliza la misma infraestructura servidora o transaccional (“de back end”).

3. **La arquitectura SOA permite la simplificación del desarrollo de soluciones mediante la utilización de estándares de la industria y capacidades comunes de industrialización.**

   La arquitectura SOA desacopla los tres componentes de una aplicación: presentación, orquestación de procesos y lógica de negocio, a la vez que estandariza la comunicación entre cada una de las capas. Todo ello favorece a que el proceso de construcción se pueda dividir y por lo tanto industrializar más fácilmente.

   Además, las empresas se pueden focalizar en los componentes de mayor valor como los procesos y externalizar o comprar el resto de componentes.

4. **La arquitectura SOA permite aislar mejor a los sistemas frente a los cambios generados por otras partes de la organización (protección de las inversiones realizadas.**

Al organizar los sistemas en módulos más pequeños (servicios) se reduce notablemente el impacto de los cambios. Por otra parte, durante las últimas décadas, las organizaciones han realizado fuertes inversiones en sus infraestructuras tecnológicas. A través de la creación de un modelo flexible que pueda reconfigurarse en función de las necesidades del negocio, la arquitectura SOA reutiliza, de un modo efectivo, los distintos sistemas tecnológicos actuales, por ejemplo, identificando la funcionalidad bajo los sistemas tecnológicos actuales y encapsulándolos en servicios que pueden ser utilizados por diferentes aplicaciones y procesos.

Al respecto, las principales compañías que ofrecen herramientas de “discovery” están reorientando sus productos para ofrecer la identificación de reglas de negocio y servicios de los sistemas actuales, para facilitar su evolución hacia SOA.

5. **La arquitectura SOA permite alinear y acercar las áreas de tecnología y negocio**

SOA cubre la brecha entre la visión del negocio y la de sistemas, estableciendo un marco de diálogo con un lenguaje común: los procesos de negocio. Las áreas de negocio se centran en la definición de los procesos de acuerdo a la estrategia y el modelo de negocio de la compañía.

El área de tecnología implementa los procesos a partir de la utilización de servicios existentes y la creación de nuevos cuando es necesario. Cuando el negocio requiere cambios en los procesos existentes, éstos se realizan de forma flexible y ágil, pues están implementados mediante tecnología estándar y servicios reutilizables. Además, por primera vez, hay una definición común de las aplicaciones: los procesos, que tanto el área de tecnología como el área de negocio comparten y entienden.

Desventajas

Se exponen las siguientes desventajas de SOA:

1. SOA requiere un cambio en las organizaciones, un alto esfuerzo. No siendo sencillo, para la mayoría de las organizaciones adoptar SOA.
2. Los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, entre otros. Esto necesariamente implica la utilización de mensajería confiable.
3. La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red, configuración, entre otros.
4. Debe manejar comunicaciones no confiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etcétera.

o **COMPONENTES SOA**

Esta arquitectura presenta un modelo de construcción sistemas distribuidos en el que la funcionalidad demandada será entregada a la aplicación a través de servicios. En la siguiente figura se muestra el esquema de la arquitectura y los elementos que podrían observarse. Como puede observarse, el esquema se encuentra dividido en 2 zonas; una que abarca el ámbito funcional de la arquitectura y otra vinculada a la calidad de servicio.

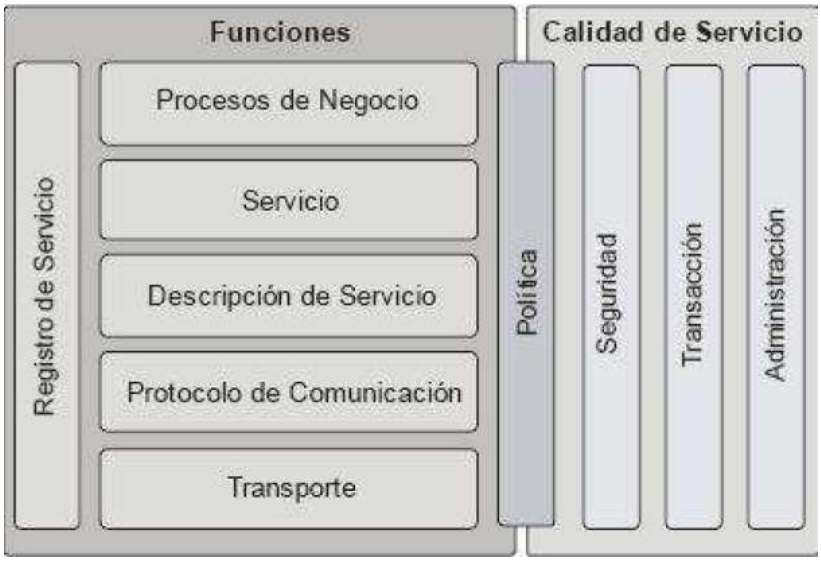


Figura II.2 Elementos de SOA

A continuación se describen los elementos representados en la figura 3:

Funciones:

- ✓ Transporte: es el mecanismo utilizado para llevar las demandas de servicio desde un consumidor de servicio hacia un proveedor de servicio, y las respuestas desde el proveedor hacia el consumidor.
- ✓ Protocolo de comunicación de servicios: es un mecanismo acordado a través del cual un proveedor de servicios y un consumidor de servicios comunican qué está siendo solicitado y qué está siendo respondido.
- ✓ Descripción de servicio: es un esquema acordado para describir qué es el servicio, cómo debe invocarse, y qué datos requiere el servicio para invocarse con éxito.
- ✓ Servicio: describe un servicio actual que está disponible para utilizar.
- ✓ Procesos de Negocio: es una colección de servicios, invocados en una secuencia particular con un conjunto específico de reglas, para satisfacer un requisito de negocio.

- ✓ Registro de Servicios: es un repositorio de descripciones de servicios y datos que pueden utilizar los proveedores de servicios para publicar sus servicios, así como los consumidores de servicios para descubrir o hallar servicios disponibles.

Calidad de Servicio:

- ✓ Política: es un conjunto de condiciones o reglas bajo las cuales un proveedor de servicio hace el servicio disponible para consumidores.
- ✓ Seguridad: es un conjunto de reglas que pueden aplicarse para la identificación, autorización y control de acceso a consumidores de servicios.
- ✓ Transacciones: es el conjunto de atributos que podrían aplicarse a un grupo de servicios para entregar un resultado consistente.
- ✓ Administración: es el conjunto de atributos que podrían aplicarse para manejar los servicios proporcionados o consumidos.

Las colaboraciones en SOA siguen el paradigma descubrir, ligar e invocar, donde un consumidor de servicios realiza la localización dinámica de un servicio consultando el registro de servicios para hallar uno que cumpla con un determinado criterio. Si el servicio existe, el registro proporciona al consumidor la interfaz de contrato y la dirección del servicio proveedor. El siguiente diagrama ilustra las entidades (roles, operaciones y artefactos) en una arquitectura orientada a servicios donde éstas colaboran.

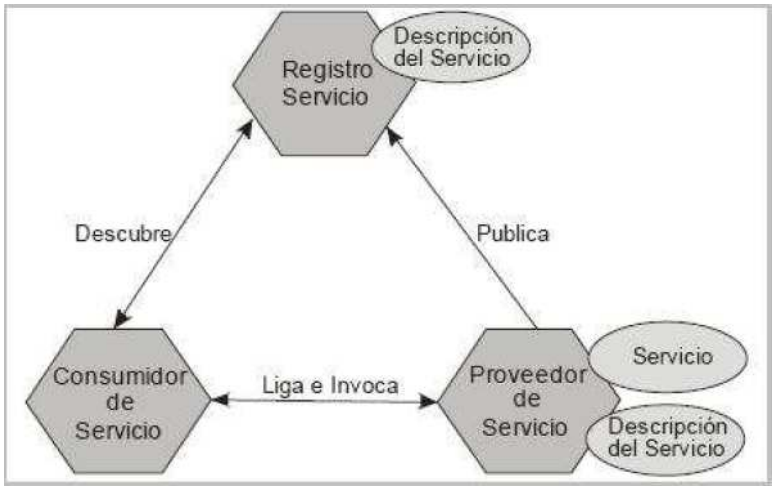


Figura II.3 Búsqueda de Servicios

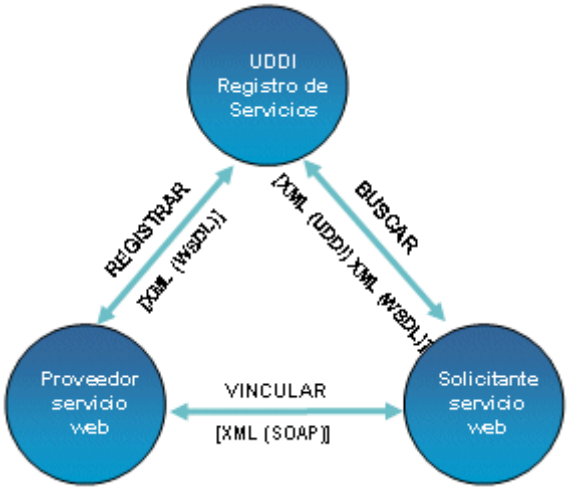


Figura II.4 Relación de entidades

Cada entidad puede tomar uno de los tres roles posibles correspondientes a consumidor, proveedor y/o registro:

- Un consumidor de servicios es una aplicación, un módulo de software u otro servicio que demanda la funcionalidad proporcionada por un servicio, y la ejecuta de acuerdo a un contrato de interfaz.
- Un proveedor de servicios es una entidad accesible a través de la red que acepta y ejecuta consultas de consumidores, y publica sus servicios y su contrato de

interfaces en el registro de servicios para que el consumidor de servicios pueda descubrir y acceder al servicio.

- Un registro de servicios es el encargado de hacer posible el descubrimiento de servicios, conteniendo un repositorio de servicios disponibles y permitiendo visualizar las interfaces de los proveedores de servicios a los consumidores interesados.

Las operaciones que pueden llevar a cabo las entidades son:

- Publicar. Para poder acceder a un servicio se debe publicar su descripción para que un consumidor pueda descubrirlo e invocarlo.
- Descubrir. Un consumidor de servicios localiza un servicio que cumpla con un cierto criterio consultando el registro de servicios.
- Ligar e Invocar. Una vez obtenida la descripción de un servicio por parte de un consumidor, éste lo invoca haciendo uso de la información presente en la descripción del servicio.

Finalmente, los artefactos en una arquitectura orientada a servicios son:

- Servicio. Un servicio que está disponible para el uso a través de una interfaz publicada y que permite ser invocado por un consumidor de servicios.

Descripción de servicio. Una descripción de servicio especifica la forma en que un consumidor de servicio interactuará con el proveedor de servicio, especificando el formato de consultas y respuestas desde el servicio. Esta descripción también puede especificar el conjunto de precondiciones, pos condiciones y/o niveles de calidad de servicio (QoS).

o **DISEÑO Y DESARROLLO DE SOA**

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios. La arquitectura orientada a servicios es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implementación. Para que un proyecto SOA tenga éxito los desarrolladores de

software deben orientarse ellos mismos a esta mentalidad de crear servicios comunes que son orquestados por clientes o middleware para implementar los procesos de negocio. El desarrollo de sistemas usando SOA requiere un compromiso con este modelo en términos de planificación, herramientas e infraestructura.

Cuando se habla de arquitectura orientada a servicios están hablando de un juego de servicios residentes en Internet o en una intranet, usando servicios web. Existen diversos estándares relacionados a los servicios web. Incluyen los siguientes:

XML

HTTP

SOAP

WSDL

UDDI

Hay que considerar, sin embargo, que un sistema SOA no necesariamente necesita utilizar estos estándares para ser "orientado a servicios" pero es altamente recomendable su uso.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar SOA utilizando cualquier tecnología basada en servicios.

o **CAPAS DEL MODELO CONCEPTUAL DE SOA**

El modelo conceptual de lo que sería una arquitectura n layer de una arquitectura orientada a servicios. A continuación se describe detalladamente que papel juega cada una de esas capas en una arquitectura SOA.

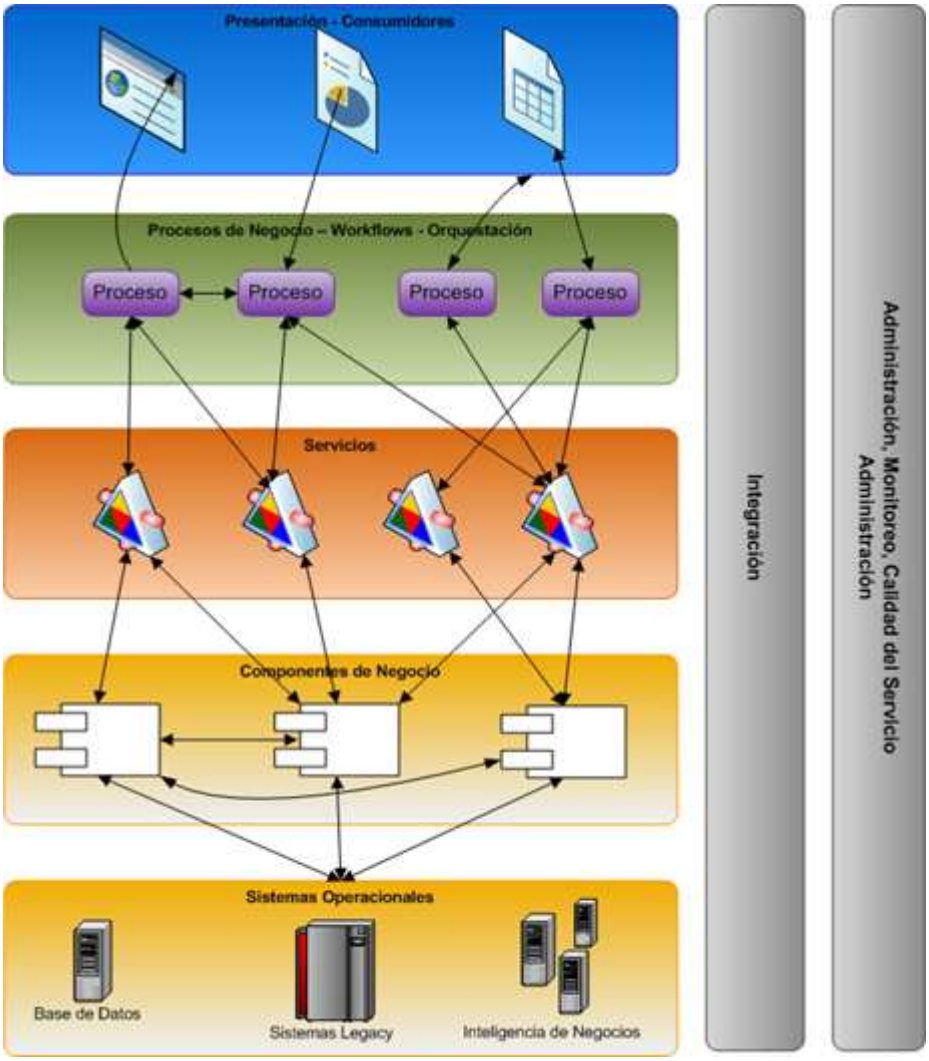


Figura II.5 Capas de SOA

En esta figura agregamos la capa de integración y la capa de Administración, Monitoreo, Calidad de Servicio y Administración. Ambas capas serán descritas en este post.

Capa 1: Sistemas Operacionales. Esta capa consiste en las aplicaciones existentes dentro de la empresa, conocidas como legacy systems. Entre estas capas podemos tener CRM´s, ERP´s, aplicaciones de BI, Orientadas a objetos, etc. Todas estas aplicaciones se integran a través de SOA.

Capa 2: Capa de Componentes: Esta capa es la que contiene los componentes que se encargan de brindar la funcionalidad que exponen los servicios. Esta capa típicamente usa tecnologías para contener los componentes que existen dentro de

esta, tales como servidores de aplicaciones, los cuales a su vez ayudan a llevar a cabo tareas como implementar componentes, a manejar el balanceo de los componente, la disponibilidad, etc.

Capa 3: Capa de Servicios: En esta capa residen los servicios que la organización decide exponer. Pueden ser descubiertos, referenciados directamente, o ser parte de una orquestación o de un servicio compuesto. Normalmente estos servicios exponen la funcionalidad de negocio a través de contratos que permiten invocar los componentes de negocio que se encuentran en la capa de componentes de la empresa. Estos contratos permiten cambiar la forma en que se llevan a cabo las tareas sin necesidad de hacer redeploy de los servicios expuestos.

Capa 4: Procesos de Negocio – Orquestación: En esta capa se exponen las orquestaciones de los servicios. Los servicios están ligados a estos workflows, y por lo tanto actúan como una sola aplicación. Aquí se utilizan herramientas visuales para construir los flujos de trabajo tales como el diseñador de orquestación de Biztalk Server, o alguna herramienta de terceros que me permita crear workflows en notación BPEL.

Capa 5: Capa de Presentación. Normalmente esta capa no forma parte de SOA, pero cada día se vuelve más relevante. Los usuarios acceden los servicios y las orquestaciones invocando desde diversas interfaces de usuario la funcionalidad que desean consumir.

Capa 6: Integración (ESB – Enterprise Service Bus). Esta capa facilita la integración de servicios a través de la introducción de un conjunto de capacidad tales como ruteo, mediación de protocolos, mecanismos de transformación, etc. Con el WSDL (Web Service Description Language) se especifica el binding, el cual implica la localización del servicio que se provee. Al mismo tiepmo, el ESB nos da la facilidad de tener independencia de la ubicación del servicio para su integración, ya que es el ESB el que al final controla el ruteo de los mensajes que le llegan para ser procesados.

Capa 7: Administración, Monitoreo y Calidad del Servicio. Esta capa nos da las características requeridas para monitorear, administrar y mantener la calidad del servicio en áreas tales como seguridad, desempeño, y disponibilidad. Se le conoce como el SOA governance.

En el siguiente post vamos a tocar el tema de los enterprise service bus, vamos a definir que es un ESB y para que nos sirve un ESB en una arquitectura orientada a servicios.

- **DIFERENCIAS CON OTRAS ARQUITECTURAS**

Al contrario de las arquitecturas orientado a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (p.ej., WSDL). La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Plataforma Java o Microsoft.NET). Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio C Sharp podría ser usado por una aplicación Java.

- **ESTRATEGIAS PARA LA IMPLANTACIÓN DE SOA**

Proporcionamos la ayuda necesaria para seleccionar la estrategia adecuada e implementar la metodología idónea para una óptima Orientación a Servicios. De esta manera se obtendrán los objetivos empresariales que se habían fijado al implementar la arquitectura SOA.

Adaptación Metodológica

Tras analizar la naturaleza del proyecto (Servicios, BPM y/o Seguridad), y partiendo de la Metodología Estándar Necesaria (“SOA Iterative Methodology”, PDCA como Metodología de Implantación de Orientación a Gestión por Procesos, y los métodos de

Implementación de Seguridad), adaptaremos, en estrecha cooperación con los departamentos de TI dicha/s Metodología/s a las necesidades del Cliente.

Definición de la Arquitectura SOA

Paralelamente al punto anterior y habiendo establecido los distintos Objetivos Empresariales, procederemos a ensamblar los elementos mínimos indispensables de gestión SOA con el fin de garantizar el éxito en la consecución de dichos objetivos.

Definición de la fase del Proyecto

Paralelamente al punto anterior y habiendo establecido los distintos Objetivos Empresariales, procederemos a ensamblar los elementos mínimos indispensables de gestión SOA con el fin de garantizar el éxito en la consecución de dichos objetivos.

Plan de Implantación

Aplicando la ventaja de los elementos Iterativos (consecución de éxitos tempranos por iteración), se procederá a realizar la planificación y el dimensionamiento de la iteración a abordar dentro de la Implementación SOA.

CAPITULO III

SERVIDORES DE APLICACIONES JavaEE

- **Definición**

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Además, Un servidor de aplicaciones es un producto basado en un componente que se encuentra en el plano medio de la arquitectura central de un servidor. Proporciona servicios de 'middleware', es decir, trabaja como un intermediario para la seguridad y el mantenimiento, además de proveer acceso a los datos.

Un servidor de aplicación maneja la mayoría de las transacciones relacionadas con la lógica y el acceso a los datos de la aplicación (esto se solía llamar 'centralización', hace algún tiempo...). La ventaja principal de un servidor de aplicaciones es la facilidad para desarrollarlas, puesto que éstas no necesitan ser programadas y en cambio, se arman a partir de módulos provistos por el servidor de aplicaciones.

El término servidor de aplicaciones se aplica a todas las plataformas, y hay muchas variaciones sobre el tema, por lo que resulta un poco ambiguo. El término se utiliza

para referirse a los servidores de aplicaciones basadas en Web, como el control de las plataformas de comercio electrónico integrado, sistemas de gestión de contenido de sitios Web y asistentes o constructores de sitios de Internet. Por esta razón, algunos los llaman también ‘servidor web’.

Uno de los ejemplos destacados es el de Sun Microsystems, plataforma JavaEE. Los servidores de aplicaciones Java se basan en la Plataforma Java ™ 2, Enterprise lEdition (JavaEE ™). JavaEE utiliza un modelo de este tipo y, en general, incluye un nivel Cliente, un nivel Medio, y un EIS. El servidor de tipo Cliente puede contener una o más aplicaciones o navegadores. La Plataforma JavaEE es del Nivel Medio y consiste en un servidor Web y un servidor EJB. (Estos servidores son también llamados "contenedores".) También podría haber sub niveles adicionales en el nivel intermedio. El nivel del Sistema Enterprise Information System (EIS, o ‘Sistema de Información Empresarial) contiene las aplicaciones existentes, archivos y bases de datos.

Para el almacenamiento de datos empresariales, la plataforma JavaEE requiere una base de datos que sea accesible a través de JDBC, SQLJ, y JDO API. La base de datos puede ser accesible desde los componentes web, desde la empresa, y desde los componentes de la aplicación cliente.

Servidor JavaEE

El estándar JavaEE permite el desarrollo de aplicaciones de empresa de una manera sencilla y eficiente. Una aplicación desarrollada con las tecnologías JavaEE permite ser desplegada en cualquier servidor de aplicaciones o servidor web que cumpla con el estándar. Un servidor de aplicaciones es una implementación de la especificación JavaEE.

La arquitectura JavaEE es la siguiente:

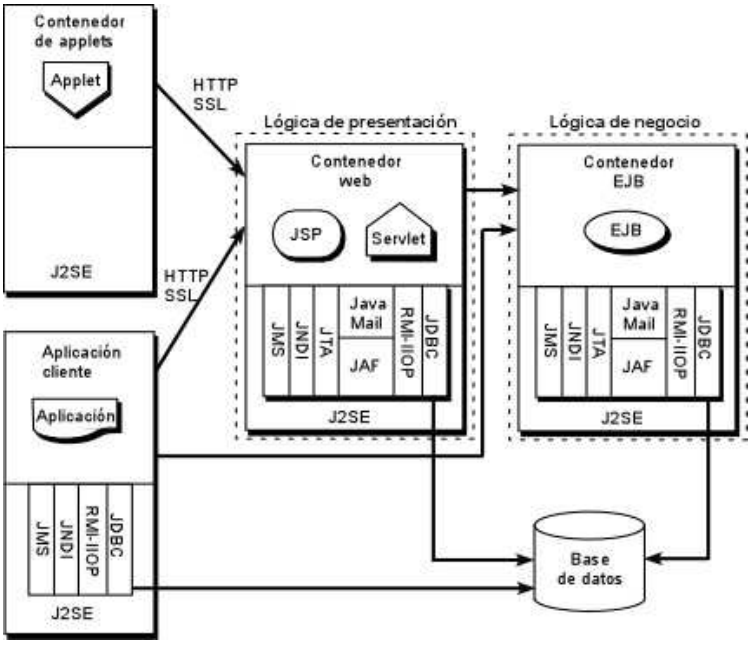


Figura III.6 Arquitectura JavaEE.

Definimos a continuación algunos de los conceptos que aparecen en la figura 6:

- **Cliente web (contenedor de applets)**: Es usualmente un navegador e interactúa con el contenedor web haciendo uso de HTTP. Recibe páginas HTML o XML y puede ejecutar applets y código JavaScript.
- **Aplicación cliente**: Son clientes que no se ejecutan dentro de un navegador y pueden utilizar cualquier tecnología para comunicarse con el contenedor web o directamente con la base de datos.
- **Contenedor web**: Es lo que comúnmente denominamos servidor web. Es la parte *visible* del servidor de aplicaciones. Utiliza los protocolos HTTP y SSL (seguro) para comunicarse.
- **Servidor de aplicaciones**: Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Es el corazón de un gran sistema distribuido.

Frente a la tradicional estructura en dos capas de un servidor web (ver Figura 7) un servidor de aplicaciones proporciona una estructura en tres capas que permite

estructurar nuestro sistema de forma más eficiente. Un concepto que debe quedar claro desde el principio es que no todas las aplicaciones de empresa necesitan un servidor de aplicaciones para funcionar. Una pequeña aplicación que acceda a una base de datos no muy compleja y que no sea distribuida probablemente no necesitará un servidor de aplicaciones, tan solo con un servidor web (usando servlets y jsp) sea suficiente.

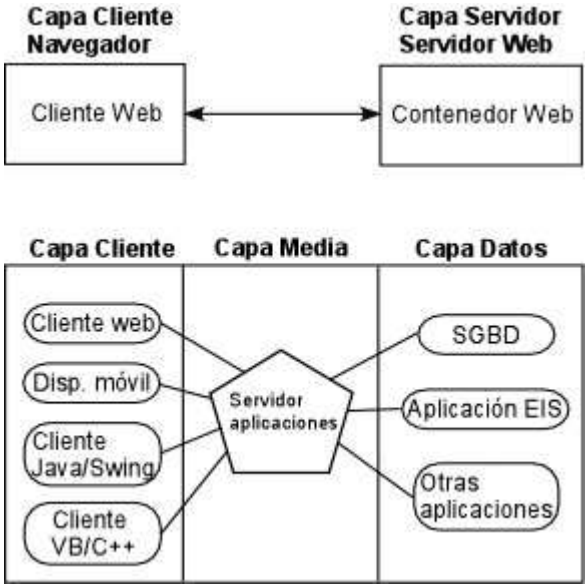


Figura III.7 Arquitectura en dos capas frente a tres capas utilizando de SA

Como hemos comentado, un servidor de aplicaciones es una implementación de la especificación JavaEE. Existen diversas implementaciones, cada una con sus propias características que la pueden hacer más atractiva en el desarrollo de un determinado sistema. Algunas de las implementaciones más utilizadas son las siguientes:

- ✓ GlassFhish
- ✓ BEA WebLogic
- ✓ JBoss
- ✓ IBM WebSphere
- ✓ Sun Netscape IPlanet
- ✓ Sun One

- ✓ Oracle IAS
- ✓ Borland AppServer

o **Características**

- ✓ Los servidores de aplicación middleware (o software de conectividad) que les permite intercomunicarse con variados servicios, para efectos de confiabilidad, seguridad, etc.
- ✓ Los servidores de aplicación también brindan a los desarrolladores una Interfaz para Programación de Aplicaciones (API), de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación web moderna.
- ✓ Los servidores de aplicación también brindan soporte a una gran variedad de estándares, tales como HTML, XML, IIOP, JDBC, SSL, etc., que les permiten su funcionamiento en ambientes web (como Internet) y la conexión a una gran variedad de fuentes de datos, sistemas y dispositivos.
- ✓ Facilidad de instalación y administración.
- ✓ Trabaja en cualquier entorno de red con protocolo TCP/IP.
- ✓ Servidor de datos.
- ✓ Servidor de páginas Web.
- ✓ Servidor de disco.
- ✓ Altísima velocidad de ejecución.
- ✓ Distribución automática de aplicaciones y del navegador.
- ✓ Mantenimiento de grupos y usuarios flexible y seguro.
- ✓ Historial de accesos y operaciones por usuario.
- ✓ Configuración de niveles de acceso por máquinas y aplicación.
- ✓ Transacciones individualizadas.
- ✓ Gestión de bloqueos automática.
- ✓ Copias de seguridad en caliente.

- ✓ Programación de tareas.
- ✓ Demonios.

Un servidor de aplicaciones le ahorrará tiempo y dinero. Para una instalación completamente nueva, escoger un servidor de aplicaciones corta sus costos a menos de la mitad. Si Ud. ya tiene equipos y está considerando actualizarlos, el ahorro es aún mayor.

o **Arquitectura de Servidor de Aplicaciones JavaEE**

La tecnología del Servidor de Aplicaciones Java es un entorno para desarrollar y ejecutar Aplicaciones Distribuidas. Estas Aplicaciones ofrecen Servicios de Nivel de Sistema como Gestión de Transacciones, Seguridad, Conectividad de Cliente y Acceso a Base de Datos. Los Enterprise Java Beans ofrecen un Modelo de Componentes que le ayudará a construir Aplicaciones de negocio con sus actuales Bases de Datos, Aplicaciones, Sistemas y Administración de Infraestructuras.

El Servidor de Aplicaciones Java ofrece: Persistencia Automática (Base de Datos y Acceso a Ficheros), Modelos de Transacción Declarativa Automáticos, Autenticación de Cliente y Control de Acceso a Nivel de Método, Gestión de Recursos para Hilos, Red y Conexiones de Base de Datos, Cacheo de Beans, Gestión de Ciclo de Vida de Beans para Crear, Encontrar y Destruir Beans, Control de Concurrencia, Propiedades de Configuración Externa de Entorno de Ejecución de Beans, Despliegue Dinámico de Beans en un Servidor en Ejecución.

La arquitectura de un servidor de aplicaciones incluye una serie de subsistemas:

- ✓ **Servidor HTTP** (también denominado servidor Web o servidor de páginas). Un ejemplo, el servidor Apache.
- ✓ **Contenedor de aplicaciones** o contenedor Servlet/JSP. Un ejemplo, Tomcat (que incluye el servicio anterior sobre páginas)

- ✓ **Contenedor Enterprise Java Beans**, que contiene aplicativos Java de interacción con bases de datos o sistemas empresariales. Un ejemplo es JBoss que contiene a los anteriores (servidor de páginas web y contenedor de aplicacione web).

Pero conviene empezar por el principio, es decir, el lenguaje básico de interconesión: el protocolo **HTTP**. Es un protocolo de aplicación, generalmente implementado sobre TCP/IP. Es un protocolo sin estado basado en solicitudes (request) y respuestas (response), que usa por defecto el puerto 8080:

- ✓ "**Basado en peticiones y respuestas**": significa que el cliente (por ejemplo un navegador) inicia siempre la conexión (por ejemplo, para pedir una página). No hay posibilidad de que el servidor realize una llamada de respuesta al cliente (retrollamada). El servidor ofrece la respueta (la página) y cierra la conexión. En la siguiente petición del cliente se abre una conexión y el ciclo vuelve e empezar: el servidor devuelve el recurso y cierra conexión.
- ✓ "**Sin estado**": el servidor cierra la conesión una vez realizada la respuesta. No se mantienen los datos asociados a la conexión. Más adelante veremos que hay una forma de persistencia de datos asociada a la "sesión".

¿Qué ocurre cuando un navegador invoca una aplicación? El cliente (el navegador) no invoca directamente el contenedor de aplicaciones, sino que llama al servidor web por medio de HTTP. **El servidor web se interpone en la solicitud** o invocación; siendo el servidor web el responsable de trasladar la solicitud al contenedor de aplicaciones.

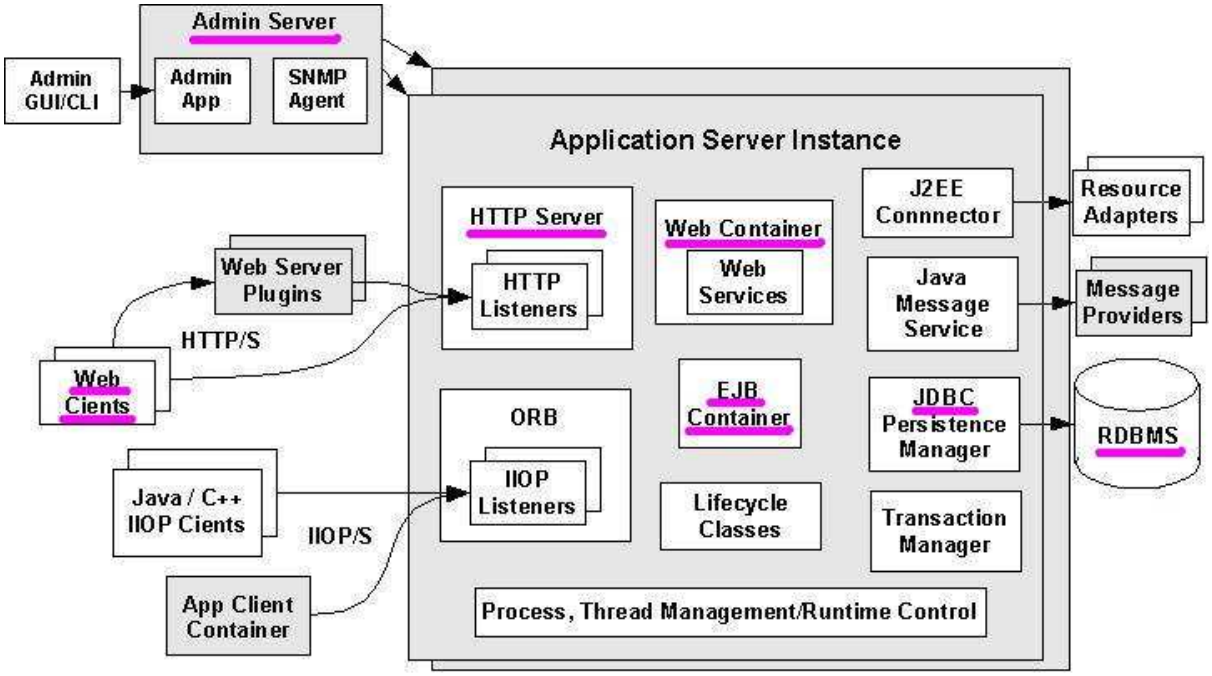


Figura III.8 Arquitectura Servidor de Aplicaciones JavaEE

Aspectos a considerar de forma síncrona:

- ✓ El **cliente** (normalmente por medio de un navegador, aunque podría ser una aplicación Swing) solicita un recurso por medio de HTTP. Para localizar el recurso al cliente especifica una URL (Uniform Resource Locator), como por ejemplo http://www.host.es/aplicacion/recurso.html. El URI (Uniform Resource Identifier) es el URL excluyendo protocolo y host. Existen diversos métodos de invocación, aunque los más comunes son POST y GET. Los veremos más adelante.
- ✓ Sobre una misma máquina podemos tener **diversas instancias** de un AS (Application Server), procurando que trabajen sobre puertos diferentes, para que no se produzcan colisiones (por defecto HTTP trabaja con 8080).
- ✓ Un servicio crucial es la capacidad de recibir peticiones HTTP, para lo cual tenemos un **HTTP Listener** (aunque puede tener listeners para otros protocolos como IIOP).

- ✓ La solicitud llega el servidor de páginas web, que tiene que descifrar si el recurso solicitado es un recurso estático o una aplicación. Si es una aplicación delega la solicitud en el **contenedor web** (contenedor Servlet/JSP). El contenedor web gestiona la localización y ejecución de Servlets y JSP, que no son más que pequeños programas. El contenedor web o contenedor Servlet/JSp recibe la solicitud. Su máquina Java (JVM) invoca al objeto Servlet/JSP, por tanto nos encontramos ante **un tipo de aplicaciones que se ejecutan en el servidor**, no en el cliente. No conviene olvidar que **un Servlet o un JSP no es más que una clase Java**. Lo más interesante en este sentido es que:
  - ✓ La JVM (generalmente) no crea una instancia de la clase por cada solicitud, sino que **con una única instancia de un Servlet/JSP se da servicio a múltiples solicitudes HTTP**. Esto hace que el consumo de recursos sea pequeño en comparación con otras opciones, como el uso de CGIs, en donde cada solicitud se resuelve en un proceso.
  - ✓ **Para cada solicitud se genera un hilo** (thread) para resolverla (pero con una única instancia de la clase, como hemos dicho).
- ✓ Un Application Server tendrá un **servidor de administración** (y normalmente un manager de la aplicación).

  Otros aspectos del contenedor web:
  - ✓ El contenedor necesita conectores que sirven de intermediarios para comunicarse con elementos externos. Los **conectores** capacitan al AS para acceder a sistemas empresariales (backends).

  Por ejemplo:
  - ✓ El **Java Message Service** ofrece conectividad con sistemas de mensajeria como MQSeries.
  - ✓ El API **JDBC** da la capacidad de gestionar bases de datos internas al AS, pero además permite ofrecer servicios como un pool de conexiones.

Es necesario una gestión de hilos, ya que será necesario controlar la situación en la que tenemos una instancia de un componente (por ejemplo, un servlet) que da respuesta a varias peticiones, donde cada petición se resuelve en un hilo.

- **Servidores de Aplicaciones JavaEE.**

Los servidores de aplicaciones desarrollados bajo el lenguaje JAVA mantienen la misma línea de administración de aplicaciones WEB, con posibilidades diversas en cuanto a capacidades de Clustering, manejo de conexiones de bases de datos, persistencia y cumplimiento del estándar JAVA EE. A continuación se inicia el estudio de los servidores propuestos y sus arquitecturas en particular.

Arquitectura GLASSFISH.

La arquitectura de GLASSFISH o Sun Java System Application Server, está basado en una estructura de clusters flexibles, además brinda el soporte para gestión remota multimáquina y multi dominio segura, lo cual pretende mejorar el grado de disponibilidad y la escalabilidad en el rendimiento del servidor de aplicaciones para sistemas de alto volumen y Servicios Web.

Componentes De La Arquitectura Glassfish

Varios elementos conforman la infraestructura en la arquitectura de GLASSFISH, elementos que se organizan y configuran permitiendo el trabajo en paralelo de varias instancias de servidor, las cuales están agrupadas en los distintos clusters, que a su vez pertenecen a nodos que en este caso representan un host distinto del servidor, todos estos elementos están gestionados por un Dominio (DAS) que ofrece un buen grado de servicio administrativo, cuyos conceptos se detallan a continuación.

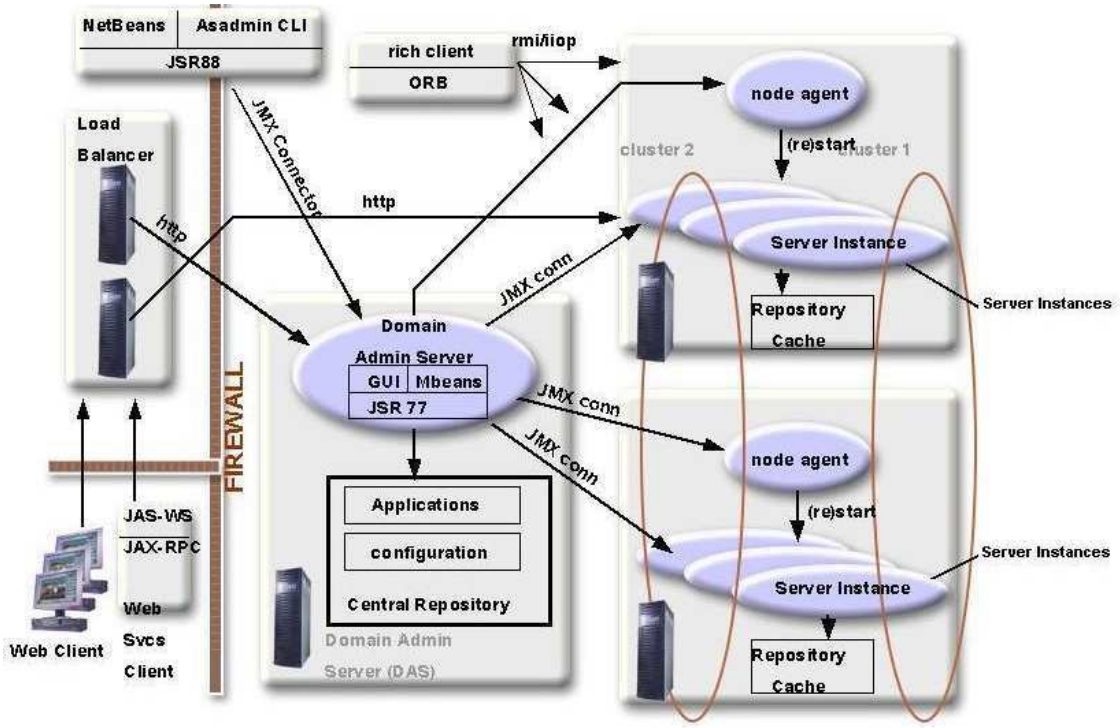


Figura III.9 Arquitectura del Servidor GlassFish

Instancia

Una instancia de servidor de aplicaciones es un motor del servidor de aplicaciones JavaEE donde se alojan las aplicaciones Java ejecutándose sobre una única Java Virtual Machine (JVM). Cada instancia funcionará como un servidor JMX MBean, es decir cuenta con los recursos administrativos para todos los servicios que brinde el servidor, puede existir dos tipos de instancias, las instancias que pertenecen a un cluster y las instancias independientes; las instancias que pertenecen a un cluster son homogéneas recibiendo los mismos recursos y configuración de su cluster padre en cambio la instancia independiente tiene su propio set de aplicaciones, recursos y configuración.[12]

[1]**Referencia:** WHITE, Larry, Kumar, Abhijit, SUN Microsystems. Clústeres de la version de Glassfish.

ttp://www.sun.com/bigadmin/hubs/multilingual/spanish/content/glassfishclust

Clúster

Un clúster es una entidad virtualizada que pertenece exactamente a un solo dominio y que contiene un grupo de instancias homogéneas trabajando en conjunto, es decir trabajan con las mismas aplicaciones, recursos e información de configuración. Un clúster es una importante entidad de administración ofreciendo a los usuarios un servicio transparente en el manejo de las instancias y sus recursos, por ejemplo una aplicación puede ser desplegada en un clúster en una operación agregada sin tener la necesidad de desplegar la aplicación en cada instancia perteneciente al clúster, el usuario no necesita estar consciente de cuantas instancias provean el servicio. Además un clúster facilita el equilibrio de la carga al ser escalable horizontalmente ya que se pueden agregar instancias adicionales de servidor. [3]

Dominio

Un dominio es un punto unificado de la administración para cualquier número de clusters o instancias del servidor así como sus configuraciones y aplicaciones que son administradas en conjunto. Este actúa como un único punto de autenticación administrativa y autorización, además en una instalación del servidor pueden agregarse múltiples dominios, actuando como mínimo un dominio que trabaje en una máquina, cada dominio puede definirse como un ente independientemente manejado, restringiendo a un desarrollador la administración de cada instancia de dominio. Por último cada dominio cuenta con un Servidor de Administración de Dominio (DAS) que se emplea para administrar instancias del servidor de aplicaciones en el dominio.[16]

DAS

---

er.jsp. 2008

[2] **Referencia:** SUN Microsystems. Application Server Commands and Concepts. http://docs.sun.com/app/docs/doc/819-3671/ablaz?a=view. 2008.

[3] **Referencia:** KALALI, Masoud; GlassFish Application Server.

http://refcardz.dzone.com/announcements/glassfish.

El Servidor de Administración de Dominio DAS (Domain Administrative Server) es una instancia del servidor de aplicaciones que administra la configuración del dominio y alberga dicha configuración en un Repositorio (ficheros en el File System). El DAS utiliza una consola para la administración del depósito central que es una interfaz basada en explorador.

Las operaciones administrativas que el DAS configura y los mantiene en el Repositorio son:

1. Configuraciones de:
    a. Dominio
    b. Instancia y su JVM
    c. Clúster y sus instancias
2. Aplicaciones desplegadas sobre cada instancia
3. Recursos (JDBC, MAIL, etc.) disponibles a cada instancia del dominio, previa asignación del administrador.

Todos los cambios de la configuración almacenada en el repositorio es replicada en las cache de configuración local de las instancias administradas por el dominio, es decir el DAS será responsable de todas las operaciones de escritura en el repositorio lo que permite una gestión centralizada de múltiples instancias ubicadas en diferentes servidores.

Otra característica a tomar en cuenta por el DAS es que la infraestructura de administración se basa en tecnología JMX (Java Management Extensions) por lo que utiliza la información de administración en forma de MBeans. Los usuarios que tiene acceso al DAS deben tener un perfil administrativo, ya sea el propio administrador del servidor de aplicaciones así como todos los clientes administrados que utilizan la API JMX. Cuando el DAS no esta en ejecución no es posible procesar operaciones administrativas, sin embargo las instancias y los clusters en el dominio pueden

funcionar normalmente con su repositorio cache y su nodo agente, es decir el DAS no tiene características de alta disponibilidad.[5][31]

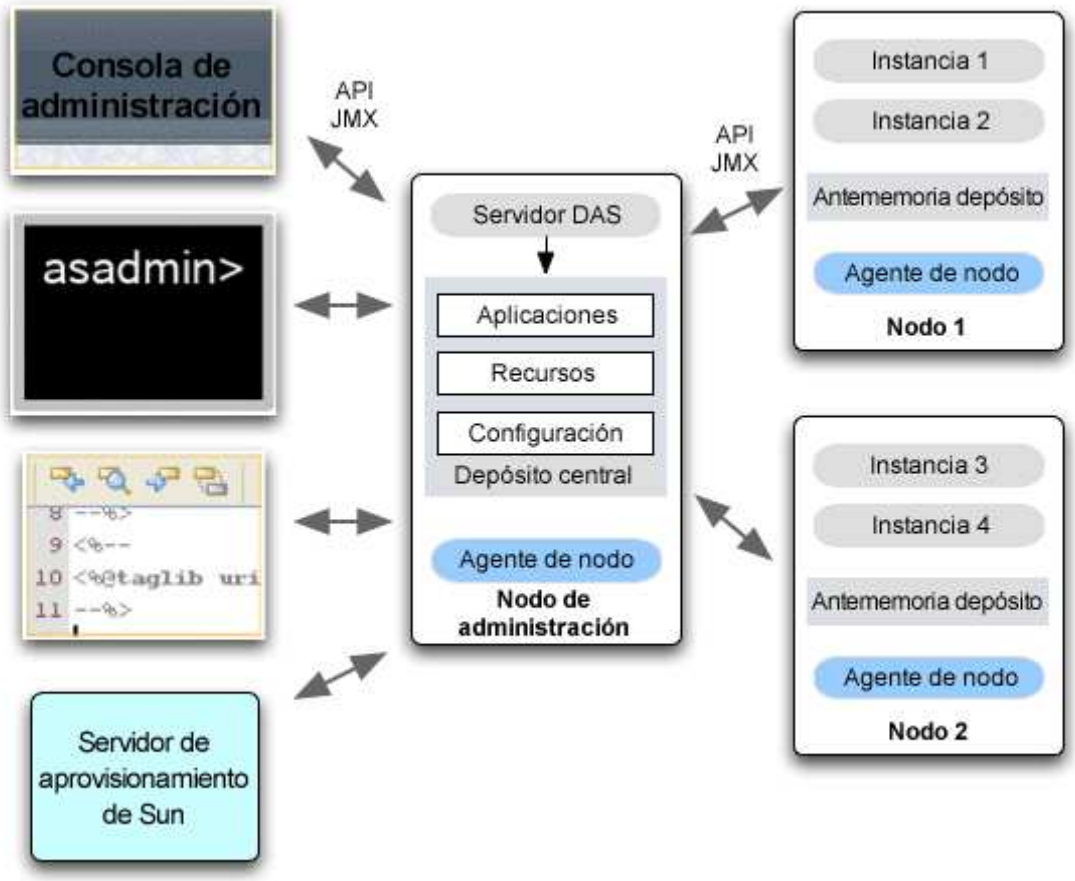


Figura III.10 Arquitectura Administrativa del DAS[4]

Nodo Agente

El nodo agente es un proceso ligero que facilita la gestión del ciclo de vida remoto de las instancias, se ejecuta en cada Host físico o nodo en el cual funciona por lo menos una instancia del servidor. El propósito del nodo agente será administrar características de las instancias como: arrancar, parar, crear y reiniciar instancias

[4] **Referencia:** WHITE, Larry, Kumar, Abhijit, SUN Microsystems. Clústeres de la versión de Glassfish.

http://www.sun.com/bigadmin/hubs/multilingual/spanish/content/glassfishcluster.jsp.

caídas, así como lectura de logs y especialmente sincronizar la configuración de todo el dominio en conexión con el DAS, por último el nodo agente corre sobre una JVM recibiendo solamente una API JMX en tiempo de ejecución.[56]

Balanceador De Carga

Es un subsistema que distribuye los requerimientos de varios servidores de aplicaciones y servidores Web dentro de las instancias del servidor es decir provee balanceo de carga para clientes HTTP, RMI/IIOP y JMS, revisando sus operaciones si es necesario y manteniendo una conexión con un host particular con el que se ha establecido una sesión.[7]

Repositorio Central Y Cache Local

Estos son utilizados para almacenar la infamación escrita por el DAS a cerca de la configuración del dominio y de las aplicaciones desplegadas en las instancias.

El repositorio central contiene toda la información de las instancias administradas bajo el mismo dominio usando un API JMX.

El cache local es un repositorio individual para cada instancia del servidor permite mantener la información de cada aplicación disponible sin utilizar el DAS dando una mayor velocidad al arranque y despliegue de las aplicaciones, el cache esta sincronizado con el repositorio central y se restaura cuando se reinicie la instancia.

Tipos De Usuarios

Para efectos de administración el servidor de aplicaciones GLASSFISH ha dividido los clientes en tres tipos.

---

[5] **Referencia**: SUN Microsystems; Sun Java Enterprise System 5 Monitoring Guide; United States.

[6]**Referencia**: SUN Microsystems; Configuring Node Agents ;

http://docs.sun.com/source/819-0215/nodeagent.html; Octubre, 2009

[7]**Referencia:** CID, Jaime; Servidores de Aplicación Arquitectura y planificación Análisis de Mercado; S/EM; Febrero, 2009

Usuarios Administrativos: Incluyen la utilidad **asadmin** CLI (Command Line Interface) y se comunican exclusivamente con el DAS vía JMX en Beans.

Usuarios Web: Se comunican con las instancias del servidor utilizando el protocolo HTTP a través de un Web browser o una invocación del servicio Web, estos clientes son lo que más utilizan el balanceador de carga

Usuarios RMI/IIOP: Este cliente reside dentro del límite del firewall del servidor se comunican con las instancias del servidor utilizando el protocolo RMI/IIOP, tiene acceso a todos los recursos JNDI definidos en el servidor, por lo tanto tiene características de administrador para ciertas actividades de configuración.

o **Arquitectura JBOSS**

El diseño del servidor de aplicaciones JBOSS utiliza componentes basados en plugins brindándole a este, características de modularidad ("conectar" o "desconectar" componentes de acuerdo a la necesidad). JBOSS utiliza como su núcleo de arquitectura el JMX (Gestión de Extensiones de Java) para poder gestionar sus componentes y también posee la característica de despliegue extensible lo que permite adicionarle componentes al JMX**.**[8]

Como punto de partida para poder entender la arquitectura JBOSS, se empezará con la introducción a JMX y los distintos módulos que interactúan con este.

JMX

JMX es un Framework que representa la "columna vertebral" que permite gestionar y monitorizar los componentes de JBOSS (módulos, contenedores, y plug ins) mediante un conjunto de APIs[9]

Mediante JMX se puede gestionar recursos de tecnología introduciendo un modelo de

[8] **Referencia:** FLEURY, Mark; STARK, Scott; NORMAN, Richards. JBoss 4.0 The Official Guide. Editorial Sams Publishing. USA. Año 2009.

[9] **Referencia:** PERRY, J Steven. Java Managment Extensions. Editorial O'Reilly. Año 2008

MBeans, los ManagedBeans (Gestión de Beans) son los objetos JMX que exponen la información manejable en forma de propiedades y operaciones necesarias para un determinado recurso.[10]

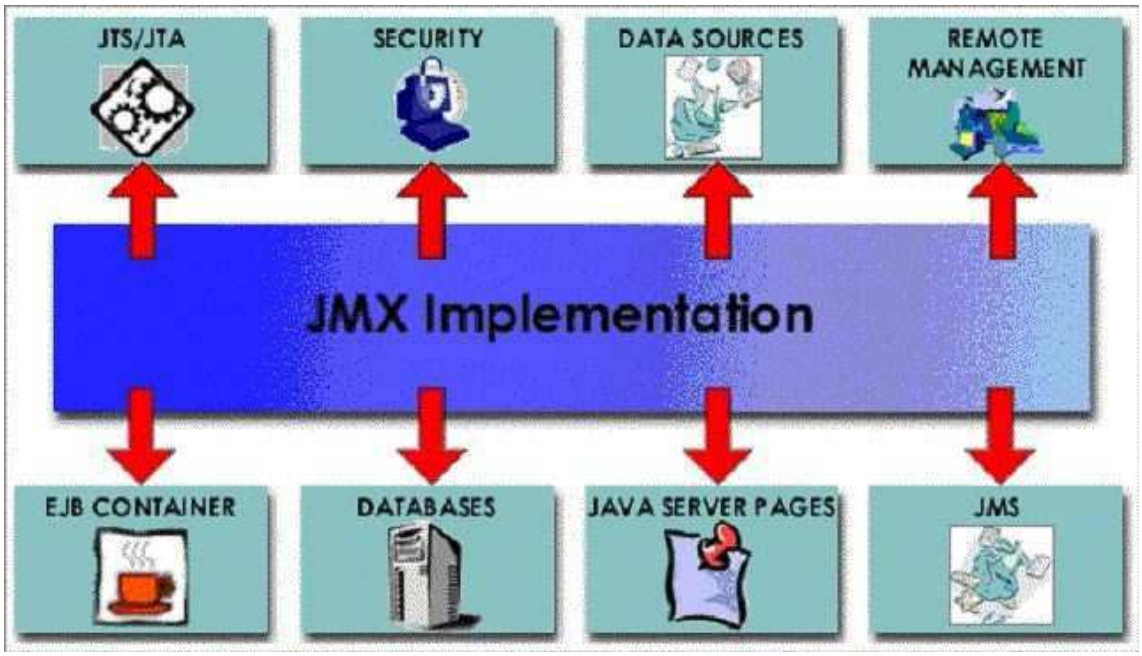


Figura III.11 Papel de integración de JMX como una columna vertebral, hacia componentes (módulos contenedores y plugins). [20]

La arquitectura JMX está conformada por tres capas o niveles las cuales son:

Nivel de Instrumentación: Este nivel o capa incluye todos los recursos o componentes que facilitan la información necesaria para la gestión de aplicaciones, en este se definen los requerimientos para implementar un recurso gestionable (aplicaciones, componentes de los servicios, dispositivos, etc.) mediante JMX.

En este nivel se presentan cuatro tipos de MBeans:

- ✓ **MBeans Estándar:** Es un java bean simple y definido estáticamente (son los más comunes dentro de JBOSS).

[10] **Referencia: P**EREIRA, Santiago. Instrumentación de componentes Java usando JMX. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jmx. Noviembre 2009

- ✓ **MBeans Dinámicos:** Estos exponen su interfaz en tiempo de ejecución.
- ✓ **MBeans Abiertos:** Estos son una extensión de los anteriores.
- ✓ **MBeans de Modelo:** Este tipo también son una extensión de los MBean Dinámicos, simplifican la instrumentación de los recursos dándole a estos un comportamiento por defecto. Los XMBeans de JBOSS son un ejemplo de este tipo de MBeans

Nivel de Agente: En este nivel se proveen los requerimientos para implementar un agente. Un agente es aquel en donde es responsable de controlar y hacer disponibles los recursos manejados en el nivel de instrumentación, como también de administrar las relaciones entre ellos.

Un componente esencial en este nivel es el MBeanServer servidor de MBeans, este MBeanServer es un registro de MBeans que permite ser accesibles a otras aplicaciones. Un MBeanServer ofrece un servicio de consulta para los MBeans, tras una consulta retorna los nombres de los MBeans. Debido a que sólo se devuelvan los nombres, todas las operaciones en todos los MBeans deben pasar por el MBeanServer. [11]

Nivel de Servicios Distribuidos: Nivel destinado a permitir una gestión cooperativa de redes de agentes y de sus recursos, en otras palabras ayuda a que las aplicaciones de administración interactúan con los agentes y sus objetos gestionados a través de adaptadores. En general, existe al menos un adaptador específico para cada protocolo de manejo o tecnología requerida para apoyar los diferentes sistemas de gestión. Permitiendo a estos componentes ampliarse para una completa aplicación de gestión.

[11]**Referencia:** PERRY, J Steven. Java Managment Extensions. Editorial O'Reilly. Año 2010

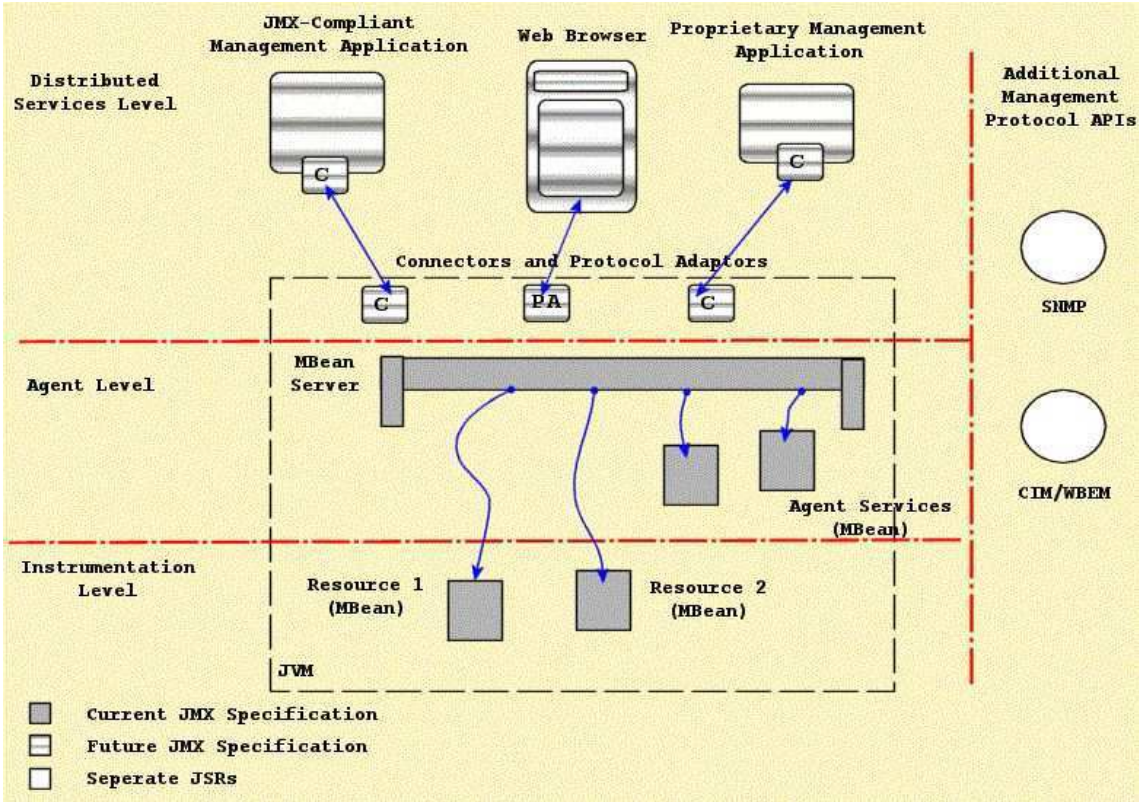


Figura III.12 Interacción entre los distintos niveles de la arquitectura JMX

Servicios Proporcionados Por JBoss

Como parte integral de la arquitectura, JBOSS cuenta con servicios que permiten la precarga de clases facilitando la comunicación entre unidades de desarrollo y servicios de la aplicación. Estos servicios son detallados a continuación.[29]

Componente JBoss NS

El componente (servicio de nombres) JBossNS es una implementación de Java Naming Directory Interface (JNDI), el cual proporciona un servicio de nombres que le permite a los usuarios asignar un nombre a un objeto o servicio para facilitar su búsqueda, además proporciona una interfaz común para servicios de directorios existentes tales como DNS, LDAP, Active Directory, RMI registry, COS registry, NIS y sistemas de archivos. El API JNDI está dividido lógicamente en un API cliente que se utiliza para acceder a los servicios de nombres, y una interfaz proveedor de servicios

(SPI) que permite al usuario crear implementaciones JNDI para el uso de los servicios de nombre.[12]

Componente JBoss TX

El componente JBoss TX está diseñado para usar cualquier gestor de transacciones que implementen el JTA (Java Transacción API). Este servicio garantiza el cumplimiento de las cuatro importantes propiedades de las transacciones las cuales son: atomicidad, consistencia, aislamiento y durabilidad (A.C.I.D.). JBossTX depende externamente del JBossNS debido a que su implementación esta ligada al manejo de nombres proporcionada por JNDI cuando este es inicializado por el servicio de administración del servidor JBOSS. [19]

Componente Contenedor EJB

El componente contenedor EJB es una arquitectura que hace hincapié en el diseño modular plugin, en la cual reside toda la lógica del negocio implementados en los EJB (Enterprise Java Beans). Los cuales proporcionan un conjunto de características o servicios como (mensajería, soporte a transacciones distribuidas, acceso a base de datos, servidores de correo, seguridad, concurrencia, etc.) todos estos aspectos clave del contenedor EJB podrán ser implementados y gestionados por versiones personalizadas de un plugin realizado por un desarrollador. Por último una gran ventaja de manejo de servicios por medio de EJBs es que estos permiten flexibilidad y también ser reutilizados. [4]

Componente JBoss MQ

El componente JBossMQ se compone de varios subsistemas que trabajan juntos para proporcionar el servicio JMS API a las aplicaciones cliente El JMS API Java Message Service Application Programming Interface, utilizado por las aplicaciones para enviar mensajes a otras aplicaciones de manera asíncrona. En modo asíncrono el emisor del

[12] **Referencia:** JBOSS.org. The JBoss 5 Application Server Guide. http://docs.jboss.org/jbossas/jboss4guide/r5/html/index.html año 2009

mensaje no tiene por qué esperar a que éste llegue a su destino. El emisor emite el mensaje y continua trabajando, el receptor lo recibe y a partir de ahí puede realizar numerosas tareas con él. Para dar una calidad de característica de fiabilidad el JMX utiliza como recurso intermediario al MOM (Messaging Oriented Middleware) quien recepta los mensajes enviados por un emisor y los redirecciona a los destinos adecuados, garantizando que el mensaje llegue al destino aunque no se encuentre disponible el receptor a través de un método de persistencia.

Componente JBoss CX

El componente JBossCX proporciona la arquitectura para manejar de manera adecuada el API JCA el cual tiene la función de integrar o conectar los EJBs que se ejecutan en el servidor de aplicaciones con sistemas externos de información conocidos como EIS (Enterprise Information System) mediante mecanismos homogéneos que da resultado a ser transparente y portable con respecto al EIS. Dichos EIS proporcionan un adaptador de recursos que sirven como intermediarios para establecer una conexión con el servidor de aplicaciones siempre y cuando soporte JCA esto en base a “contratos” establecidos por la arquitectura JCA que estandarizan la conexión estos tipos de contratos son de Connection Management, Transaction Management y Security, los cuales proporcionan escalabilidad, acceso transaccional y seguridad respectivamente. En forma general los conectores JCA han sido elaborados para resolver el acceso a sistemas legados en una modalidad sincrónica (petición/respuesta), lo cual se refleja en la mayoría de los conectores existentes.

Componente JBoss SX

El componente JBossSX diseñado para el control de acceso y autorización de usuarios mediante el API JAAS, el cual es un conjunto de paquetes con servicios para autenticar y controlar el acceso de una manera centralizada (en un descriptor.) además la característica primordial que posee JAAS es la inclusión de un dominio de

seguridad que su principal utilidad es la de servir como una especie de JMS (Java Message Service). El objetivo principal de JAAS es delegar el mecanismo de autenticación y autorización en el servidor de aplicaciones mediante la asignación de roles a los usuarios de la aplicación J2EE, dichos roles de asignación no deben ser codificados sino mas bien especificados en un archivo XML llamado descriptor de despliegue (deployment descriptor) es así como JAAS maneja el acceso a recursos y servicios de acuerdo al perfil configurado por el propio usuario.[19]

Componente Contenedor Web

El componente contenedor WEB es una implementación que especifica un entorno de ejecución para componentes Web que son capaces de manejar conexiones HTTP, es decir maneja la ejecución de Servlets y páginas JSP (Java Server Pages) lo que permite que estas tecnologías ayuden a la creación de páginas Web dinámicas es decir, páginas web que se generan en el servidor en base a la lógica de negocio.[27]

La forma de comunicación entre el contenedor web y el usuario está basado en un sistema de peticiones y respuestas lo que significa que el usuario inicia siempre una conexión y el servidor retorna una respuesta y cierra la conexión una vez realizada la respuesta, dependiendo de los recursos que el usuario maneje, el contenedor utilizará conectores que sirven de intermediarios entre la comunicación con elementos externos como el uso de otros servicios ya sea para sistema de mensajería o gestión de base de datos, etc.

Deployment

El servicio de DEPLOYMENT apoya los despliegues de los EJB jars, wars y ears. Este vigila las urls para los archivos J2EE y despliega los archivos tal como aparecen o cambian sin la necesidad de reiniciar el servidor JBoss si se encuentra en ejecución.

Otros Servicios Proporcionados Por JBoss

El servidor de aplicaciones JBOSS proporciona nuevos servicios que se adaptan a las necesidades empresariales particulares de ciertos usuarios, por lo cual estos paquetes

no están incluidos en una instalación “típica” entre estos servicios se detalla a breves rasgos los siguientes.

- ✓ **Hibernate**

Es un servicio de persistencia que proporciona una simple, pero poderosa, alternativa a los “Beans” estándares. Facilita la creación de clases de persistencia utilizando el lenguaje Java, incluyendo la asociación, herencia, polimorfismo y composición. Hibernate es muy simple, pero cuando se ejecuta en JBoss, puede optar por desplegar su aplicación como archivo Hibernate, llamado archivo HAR.

- ✓ **Clustering**

Para mejorar el funcionamiento de una aplicación si esta abarca una dimensión considerada JBOSS permite el manejo de nodos los cuales conforman un cluster para que dicha aplicación se ejecute en paralelo. Cada nodo es un servidor JBOSS físico lo que garantiza la recuperación de la aplicación si uno de los nodos falla.[20]

- ✓ **JGroups**

Proporciona el servicio peer to peer para que se puedan llevar a cabo comunicaciones entre nodos que formen parte de un mismo cluster.

- ✓ **JBoss Cache**

Es un servicio Diseñado para almacenar en caché los objetos Java a los que se accede más frecuentemente de manera que se aumente el rendimiento de las aplicaciones. Eliminando accesos innecesarios a la base de datos, reduce el tráfico de red e incrementa la escalabilidad de las aplicaciones.

- ✓ **JBoss Portal**

Es una Plataforma de código abierto para albergar y servir una interfaz de portales Web, publicando y gestionando el contenido así como adaptando el aspecto de la presentación.

- ✓ **JBoss AOP**

Este servicio AOP (Aspect Oriented Programming) es una innovación relevante en JBOSS ya que soporta el desarrollo orientado a aspectos, el **aspecto** es un comportamiento común en el que se identifica claramente métodos, clases y objetos, por lo que permite mayor modularidad que en la programación orientada a objetos y la reutilización de código para un mejor rendimiento en la aplicación.

- o **Arquitectura WEBLOGIC**

WebLogic Server es un servidor de aplicaciones: una plataforma para aplicaciones empresariales multi capa distribuidas. WebLogic Server centraliza los servicios de aplicación como funciones de servidor web, componentes del negocio, y acceso a los sistemas "backend" de la empresa. Utiliza tecnologías como el almacenamiento en memoria inmediata y almacenes de conexiones para mejorar la utilización de recursos y el funcionamiento de la aplicación. WebLogic Server también proporciona facilidades a nivel de seguridad empresarial y una administración poderosa.

WebLogic Server funciona en la capa media (o capa "n") de una arquitectura multi capa. Una arquitectura multi capa determina dónde se ejecutan los componentes software que crean un sistema de cálculo en relación unos con otros y al hardware, la red y los usuarios.

WebLogic Server implementa J2EE, el estándar para la empresa de Java. Java es un lenguaje de programación, seguro ante la red, orientado a objetos, y J2EE incluye la tecnología de componentes para desarrollar objetos distribuidos. Estas funciones agregan una segunda dimensión arquitectura del servidor de aplicaciones WebLogic Server un capa de lógica de aplicación, con cada capa desplegada selectivamente entre las tecnologías J2EE de WebLogic Server.

Capas de Componentes Software

Los componentes software de una arquitectura multi capa constan de tres capas:

- ✓ La capa del cliente contiene los programas ejecutados por los usuarios, incluyendo navegadores Web y programas de aplicaciones de red. Estos programas se pueden escribir virtualmente en cualquier lenguaje de programación.
- ✓ La capa media contiene el servidor WebLogic y otros servidores que son direccionados directamente por los clientes, como servidores web existentes o servidores proxy.
- ✓ La capa backend contiene recursos de empresa, como sistemas de base de datos, aplicaciones de unidad central y legales, y aplicaciones de plannings de recursos de empresa empaquetados (ERP).

Las aplicaciones del cliente tienen acceso al servidor WebLogic directamente, o a través de un servidor web o un proxy. El servidor WebLogic conecta con serviciosbackend por cuenta de los clientes, pero los clientes no tienen acceso directamente a los servicios backend.

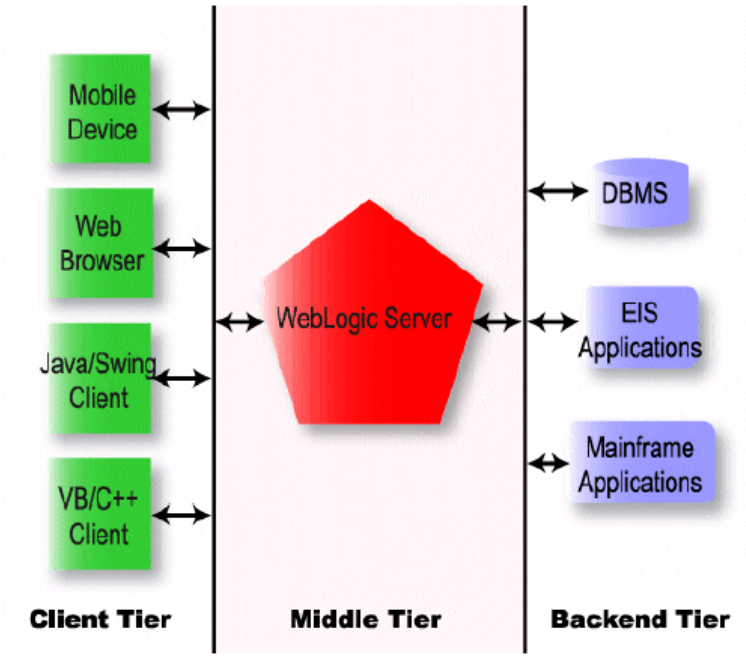


Figura III.13 Capas de la arquitectura del servidor WEBLOGIC

Componentes de la Capa Cliente

Los clientes del servidor WebLogic utilizan interfaces estándares para acceder a servicios del servidor WebLogic. El servidor WebLogic tiene una completa funcionalidad de servidor web, así que un navegador web puede solicitar páginas al servidor WebLogic usando el protocolo estándar de la Web, HTTP. Los servlets de WebLogic Server y las JavaServer Pages (JSPs) producen páginas Web dinámicas, personalizadas requeridas para las aplicaciones avanzadas de comercio electrónico. Los programas del cliente escritos en Java pueden incluir interfaces gráficos de usuario altamente interactivos construidos con las clases de Java Swing. También se puede tener acceso a servicios del servidor WebLogic usando los APIs estándar del J2EE.

Componentes de la Capa Media

La capa media incluye el servidor WebLogic y otros servidores Web, cortafuegos, y servidores proxy que median en el tráfico entre los clientes y el servidor WebLogic.

La opciónCluster del servidor WebLogic permite que distribuyamos peticiones de cliente y servicios backend entre varios servidores WebLogic cooperantes. Los programas en la capa del cliente acceder al cluster como si fuera un solo servidor WebLogic. Cuando la carga de trabajo aumenta, podemos agregar otros servidores WebLogic al cluster para compartir el trabajo. El cluster utiliza un algoritmo de balance de capa seleccionable para elegir el servidor WebLogic el cluster que es capaz de manejar la petición.

Cuando una petición falla, otro servidor WebLogic que proporciona el servicio solicitado puede asumir el control. Los fallos son transparentes siempre que sea posible, lo que reduce al mínimo la cantidad de código que se debe escribir para recuperar incidentes. Por ejemplo, el estado de la sesión de un servlet se puede replicar en un servidor secundario WebLogic de modo que si el servidor WebLogic que

está manejando una petición falla, la sesión del cliente se pueda reanudar de forma ininterrumpida desde el servidor secundario.

Todos los servicios de WebLogic, EJB, JMS, JDBC, y RMI están implementados con capacidades de clustering.

Componentes de la Capa Backend

La capa backend contiene los servicios que son accesibles a los clientes sólo a través del servidor WebLogic. Las aplicaciones en la capa backend tienden a ser los recursos más valiosos y de misiones críticas para empresa. El servidor WebLogic los protege de accesos directos de usuarios finales. Con tecnologías tales como almacenes de conexiones y caches, el servidor WebLogic utiliza eficientemente los recursos backend y mejora la respuesta de la aplicación.

Los servicios backend incluyen bases de datos, sistemas de hojas de operación (planning) de recursos de la empresa (ERP), aplicaciones mainframe, aplicaciones legales de la empresa, y monitores de transacciones. Las aplicaciones existentes de la empresa se pueden integrar en la capabackend usando la especificación de configuración del conector Java (JCA) de Sun Microsystems. El servidor WebLogic hace fácil agregar un interface Web a una aplicación backend integrada. Un sistema de control de base de datos es el servicio backend más común, requerido por casi todas las aplicaciones del servidor WebLogic. WebLogic EJB y WebLogic JMS normalmente almacena datos persistentes en una base de datos en la capa backend.

o **Arquitectura WEBSPHERE**

WebSphere maneja también varios conjuntos de procesamiento llamados nodos, estos se almacenan de manera lógica en Celdas. Estas celdas pueden contener un único nodo en el cual los componentes de software se encuentran instalados o múltiples nodos en los cuales los componentes de software se encuentran distribuidos.

Un celda típica de WebSphere contiene componentes de software que pueden se instalados en un nodo o distribuidos sobre múltiples nodos para propósitos de mantenimiento y escalabilidad. La celda incluye lo siguiente:

- ✓ Un servidor web que provee servicios HTTP
- ✓ Una base de datos con fines de almacenamiento.
- ✓ El servidor de Aplicaciones WebSphere.

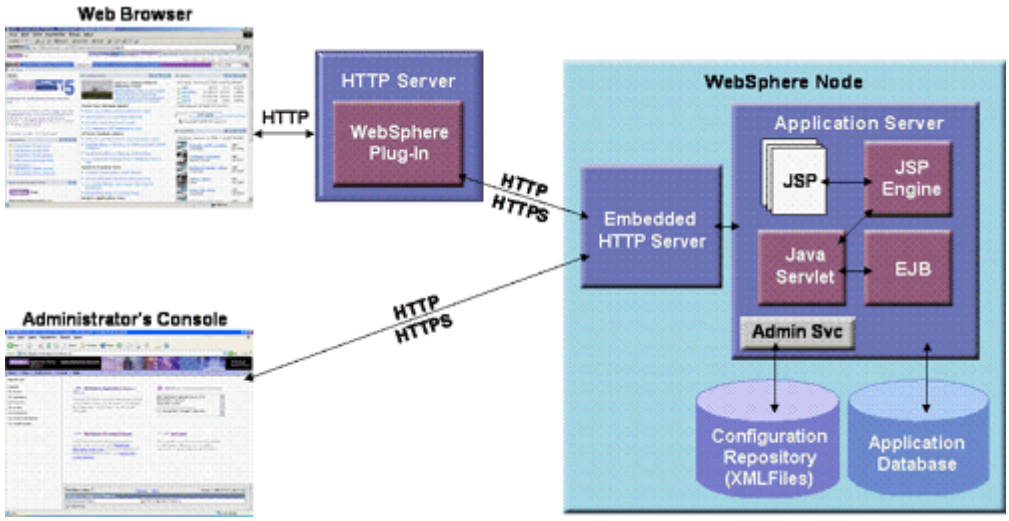


Figura III.14 Arquitectura WebSphere con un único nodo

Arquitectura Interna de WebSphere

La estructura interna de WAS es muy compleja y amplia, por esta razón se exponen aquí las partes importantes que nos ayuden a establecer un conocimiento global que nos permita cumplir con el objetivo de comparar con las otras infraestructuras expuestas anteriormente.

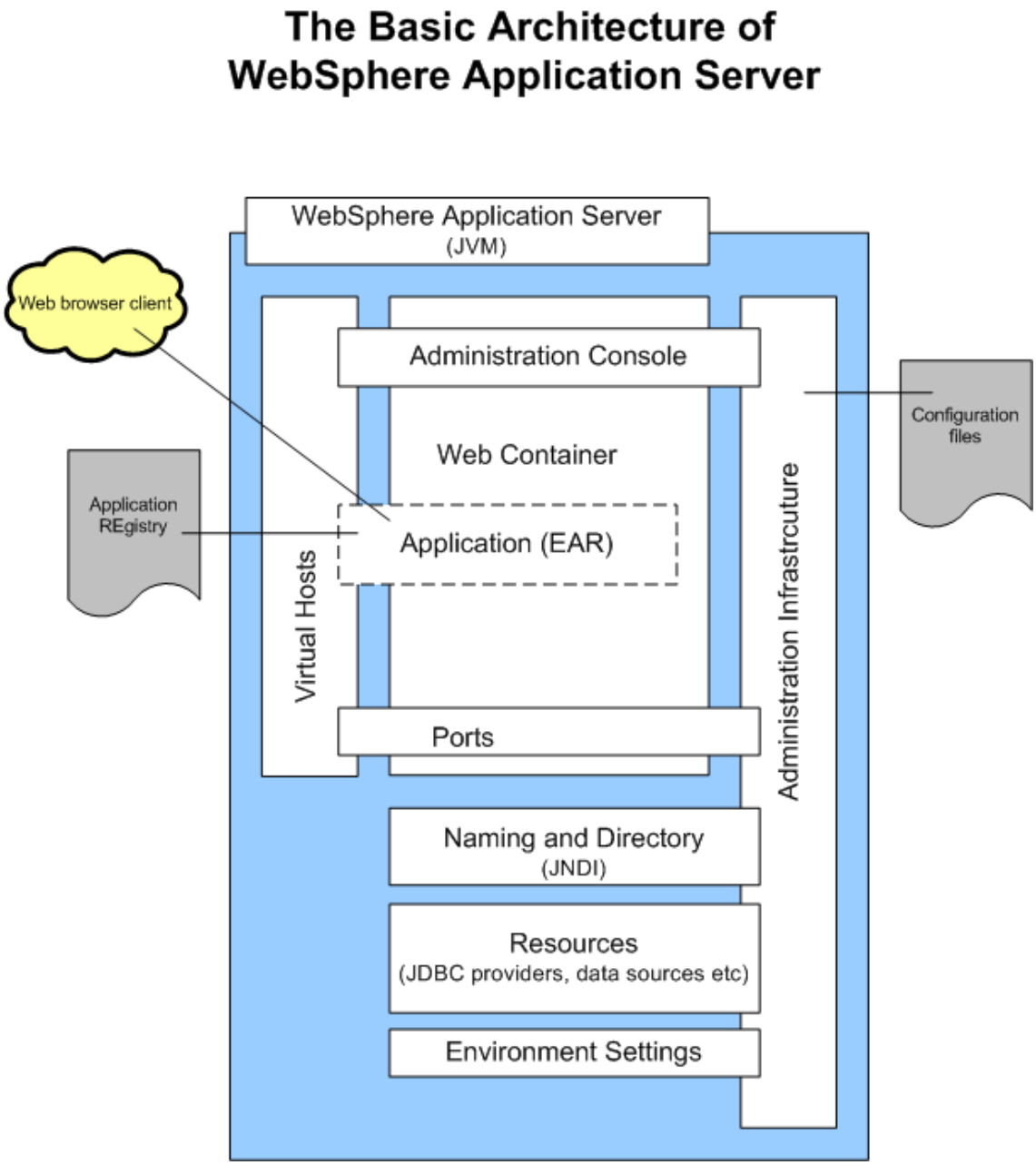


Figura III.15 Arquitectura Básica de WAS

Recalcando que el código base de WAS es el mismo para todos los sistemas operativos (plataformas), se indica que las aplicaciones Java también se pueden implementar en las distintas plataformas sin cambios en el código, siempre y cuando se utilice la misma versión de WAS.

JVM

Todos los servidores WAS son esencialmente máquinas virtuales de Java, por esta razón IBM ha puesto en marcha el modelo JAVAEE maximizando la especificación del estándar y proporcionando muchas mejoras en la creación de Servicios Web.

Contenedor WEB

Un tipo común de aplicaciones de negocio son las aplicaciones Web. El contenedor web de WAS es esencialmente un servidor web basado en JAVA incluido internamente en la JVM del servidor de aplicaciones, el cual brinda el acceso a la aplicación web para el navegador del cliente.

Hosts Virtuales

Un host virtual es un elemento de configuración requerido por el contenedor WEB para recibir las peticiones HTTP. En la mayoría de tecnologías WEB de servidor, un solo computador se requiere para alojar múltiples aplicaciones y estas aparecen en el mundo como si se tratase de varios computadores.

Los recursos que se asocian a un host virtual en particular están diseñados para no compartir datos con los recursos que pertenecen a otra host virtual, incluso si las host virtuales comparten la misma máquina física.

A cada host virtual se le asigna un nombre lógico y varios alias DNS por los cuales se le ubicará en la red.

Por defecto durante la instalación se crean dos alias para los Hosts. El primero es para la consola de administración llamado “admin_host” y el segundo denominado “default_host” el cual es asignado como la base del dominio virtual por defecto para todas las aplicaciones desplegadas, a menos que se reemplace este durante la fase de despliegue. Todas las aplicaciones alojadas deben estar asignadas a un host virtual, de lo contrario los clientes no podrán tener acceso a estas.

Variables de Entorno

WebSphere usa las variables de entorno de JAVA para controlar la configuración y las propiedades relacionadas al entorno del Servidor. Las variables de WebSphere son usadas para configurar los paths del producto, tales como la ubicación del driver para la base de datos, por ejemplo ORACLE_JDBC_DRIVER_PATH, y demás valores del entorno requerido internamente por los servicios de WebSphere y las aplicaciones que puedan estar alojadas en este.

Recursos

Los datos de configuración se almacenan en archivos XML bajo el repositorio subyacente de WAS. La definición de los recursos es un parte fundamental de la administración JAVAEE. La lógica de una aplicación puede variar en función de los requerimientos de negocio y estos se pueden personalizar mediante los diferentes tipos de recurso de aplicación. Se enumera los recursos más comúnmente utilizados:

Tabla III.1 Recursos de Aplicación WAS

Tipos de Recursos	Descripción
JDBC (Java Database Connectivity)	Se utiliza para definir los proveedores y fuentes de datos.
URL Providers	Se utiliza para definir los end points para los servicios externos, por ejemplo los servicios WEB.
JMS Providers	Se utiliza para definir la configuración de mensajería de Java Message Service, MQ connection factories, colas de destinos, etc.
Mail Providers	Habilita la funcionalidad para las aplicaciones de enviar y recibir correo electrónico, típicamente usando el protocolo SMTP.

JNDI

Java Naming and Directory Interface (JNDI) o Interfaz de Directorio de Nombres JAVA se emplea en la elaboración de aplicaciones portables. JNDI es un API esencial para

un directorio de servicios que permite a las aplicaciones JAVA la búsqueda de objetos y datos mediante su nombre. JNDI es un servicio de búsqueda donde cada recurso posee un único nombre. Las operaciones de nombrado, tales como búsquedas y enlaces se ejecutan en contextos. Todas las operaciones de nombrado comienzan con la obtención del contexto inicial. Este contexto inicial se constituye en el punto de inicio del espacio de nombres. Las aplicaciones usan búsquedas JNDI para encontrar un recurso usando una convención conocida de nombres. Los administradores pueden reemplazar los recursos de la aplicación sin necesidad de un cambio o reconfiguración en el código de la misma. Este nivel de abstracción proporcionado por JNDI es fundamental y necesario para el correcto uso de las aplicaciones WebSphere.

Tipos de Archivo de Aplicación

Existen tres tipos de archivos de aplicación con los que se trabaja en JAVA. Dos de estos pueden ser instalados mediante el proceso de despliegue de WebSphere. Uno es conocido como el archivo EAR y al segundo se lo conoce como el archivo WAR. El tercero, el archivo JAR (frecuentemente con código común reutilizable) se encuentra contenido en un WAR o EAR. La explicación de los tipos de archivos se muestra en la siguiente tabla:

Tabla III.2 Tipos de Archivo de Aplicación Java

Tipo de Archivo	Descripción
Archivo JAR	El archivo JAR (o Java ARchive) se usa para organizar varios archivos en uno solo. La distribución física interna actual es muy similar a un archivo ZIP. El JAR se utiliza generalmente para distribuir las clases java y los metadatos asociados. En las aplicaciones JAVAEE el archivo JAR contiene código de utilidades, librerías compartidas y EJB. Un EJB es un paquete del lado del servidor que encapsula la lógica de negocio de una aplicación y es una de las tantas APIs que brinda la plataforma JAVAEE con su propia especificación.
	El Enterprise Archive o Archivo empresarial representa una

Archivo EAR	aplicación JAVAEE que puede ser desplegada en el Servidor WebSphere. Los archivos EAR son archivos estándar de java (JAR) y tienen la extensión *.ear.* Este archivo contiene lo siguiente: ✓ Uno o más módulos WEB empaquetados en archivos WAR. ✓ Uno o más módulos EJB empaquetados en archivos JAR. ✓ Uno o más módulos de aplicaciones cliente. ✓ Archivos JAR adicionales requeridos por la aplicación. ✓ Cualquier combinación de las anteriores. Los archivos EAR también contienen un descriptor de despliegue (un XML llamado *aplication.xml*) que describe el contenido de la aplicación y contiene instrucciones para toda la aplicación, tales como configuraciones de seguridad que se utilizarán en el entorno en tiempo de ejecución.
Archivo WAR	Un archivo WAR (Web application) es esencialmente un JAR usado para encapsular una colección de Java Server Pages(JSP), servlets, clases Java, HTML y otros archivos relacionados entre los cuales se puede incluir XML y otros tipos dependiendo de la tecnología web utilizada. ✓ Los Servlets pueden generar contenido dinámico para páginas Web, esto se consigue mediante el procesamiento dinámico del lado del servidor y mediante el acceso a bases de datos. ✓ Los archivos JSP pueden ser usados para separar el código HTML de la lógica de negocio en las páginas Web. Esencialmente, estos también pueden generar páginas dinámicas, sin embargo estos emplean Java Beans (clases) que contienen la lógica específica y detallada del lado del servidor. Un archivo WAR también tiene su propio descriptor de despliegue llamado "*web.xml*", que se utiliza para configurar el archivo WAR y puede contener instrucciones para asignación de recursos y seguridad.

CAPITULO IV

IMPLEMENTANDO SOA CON JavaEE

- **Introducción**

Los servicios web son componentes software con estas características distintivas para el programador:

- ✓ Son accesibles medinte el protocolo SOAP ("Simple Object Access Protocol").
- ✓ Su interfaz se describe mediante un documento WSDL ("Web Services Description Language" o "Lenguaje de Descripción de Servicios Web").

SOAP es un protocolo de comunicaciones para paso de mensajes XML, fundamento sobre el cual se sustentan los servicios web. SOAP permite el envío de mensajes XML entre aplicaciones. Estos mensajes son unidireccionales, pero todas las aplicaciones pueden ser a la vez emisoras o receptoras. Los mensajes SOAP sirven para muchos propósitos, como, entre otros, esquemas de petición y respuesta, notificaciones o mensajería asíncrona.

SOAP es un protocolo de alto nivel, que define la estructura del mensaje y ciertas reglas básicas para su procesamiento, y es totalmente independiente del protocolo de transporte. Esto permite que los mensajes SOAP sean intercambiados mediante HTTP, SMTP, etc. En estos momentos, HTTP es el más utilizado.

WSDL es un estándar que describe servicios web mediante un documento XML. Este documento proporciona a las aplicaciones la información requerida para acceder a un servicio web. El documento ofrece una descripción el objetivo del servicio web, sus mecanismos de comunicación, ubicación, etc.

o **Servicios Web (Web Services)**

Definición 1:

Un Web Service es un componente de software que se comunica con otras aplicaciones codificando los mensaje en XML[13] y enviando estos mensaje a través de protocolos estándares de Internet tales como el Hypertext Transfer Protocol (HTTP). Intuitivamente un Web Service es similar a un sitio web que no cuenta con un interfaz de usuario y que da servicio a las aplicaciones en vez de a las personas. Un Web Service, en vez de obtener solicitudes desde el navegador y retornar páginas web como respuesta, lo que hace es recibir solicitudes a través de un mensaje formateado en XML desde una aplicación, realiza una tarea y devuelve un mensaje de respuesta también formateado en XML.[14]

Definición 2:

Un Web Service un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.[15]

¿Para qué sirven?

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica

[13] **Nota**: XML.- Extensible Markup Language.

[14] **Referencia:** tomada de. http://www.desarrolloweb.com/articulos/1545.php. XML Web Services

[15] **Referencia:** tomada de. http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb. Guía Breve de Servicios Web

al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

Funcionamiento de los Servicios Web

El siguiente gráfico muestra cómo interactúa un conjunto de Servicios Web:

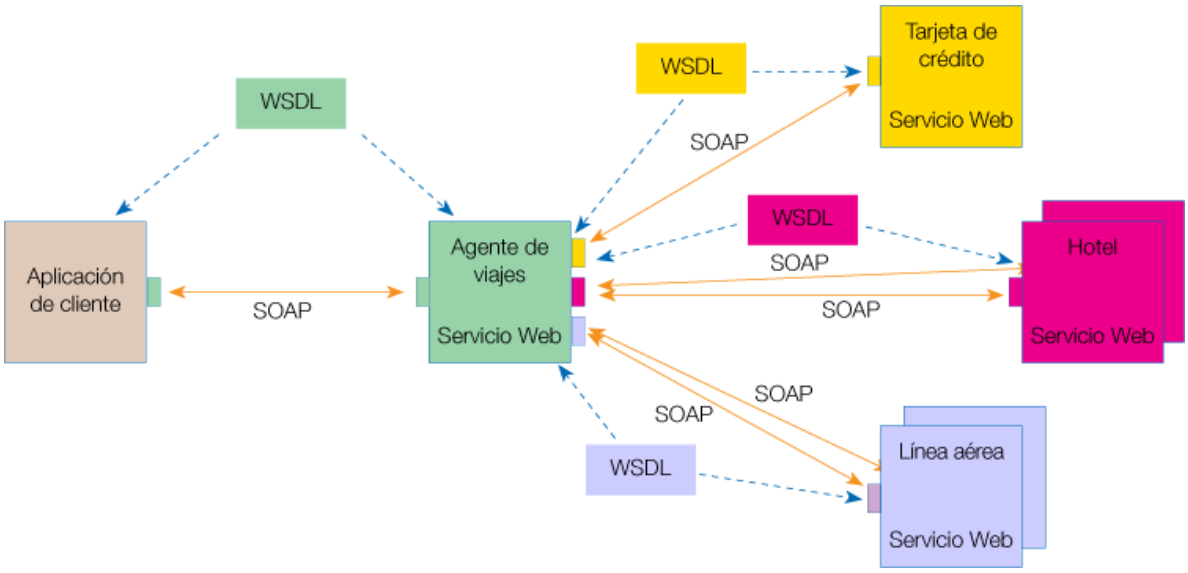


Figura IV.16 Funcionamiento de los Servicios WEB

En la figura anterior, un usuario (Cliente Servicio Web), a través de una aplicación, solicita información sobre un viaje que desea realizar haciendo una petición a una agencia de viajes que ofrece sus servicios a través de Internet. La agencia de viajes ofrecerá a su cliente (usuario) la información requerida. Para proporcionar al cliente la información que necesita, esta agencia de viajes solicita a su vez información a otros recursos (Servicios Web) en relación con el hotel y la compañía aérea. La agencia de viajes obtendrá información de estos recursos, lo que la convierte a su vez en cliente de esos otros Servicios Web que le van a proporcionar la información solicitada sobre el hotel y la línea aérea. Por último, el usuario realizará el pago del viaje a través de la agencia de viajes que servirá de intermediario entre el usuario y el servicio Web que gestionará el pago.

Requisitos de un Web Service

Interoperabilidad. Un servicio remoto debe permitir su utilización por clientes de otras plataformas.

Amigabilidad con Internet: La solución debe poder funcionar para soportar clientes que accedan a los servicios remotos desde internet.

Interfaces fuertemente tipadas. No debería haber ambigüedad acerca del tipo de dato enviado y recibido desde un servicio remoto. Más aún, los tipos de datos definidos en el servicio remoto deben poderse corresponder razonablemente bien con los tipos de datos de la mayoría de los lenguaje de programación procedimentales.

Posibilidad de aprovechar los estándares de Internet existentes. La implementación del servicio remoto debería aprovechar estándares de Internet existentes tanto como sea posible y evitar reinventar soluciones a problema que ya se han resuelto. Una solución construida sobre un estándar de Internet ampliamente adoptado puede aprovechar conjuntos de herramientas y productos existentes creados para dicha tecnología.

Soporte para cualquier lenguaje. La solución no debería ligarse a un lenguaje de programación particular Java RMI, por ejemplo, esta ligada completamente a lenguaje Java. Sería muy difícil invocar funcionalidad de un objeto Java remoto desde Visual Basic o PERL. Un cliente debería ser capaz de implementar un nuevo servicio Web existente independientemente del lenguaje de programación en el que se halla escrito el cliente

Soporte para cualquier infraestructura de componente distribuida. La solución no debe estar fuertemente ligada a una infraestructura de componentes en particular. De hecho, no se bebería requerir el comprar, instalar o mantener una infraestructura de objetos distribuidos, solo construir un nuevo servicio remoto utilizar un servicio existente. Los protocolos subyacentes deberían proporcionar un nivel base de

comunicación entre infraestructura de objeto distribuidos existentes tales como DCOM y CORBA.

Tecnologías que Intervienen

Para el proceso anterior intervienen una serie de tecnologías que hacen posible el acceso a la información, entre los cuales tenemos.

SOAP (Simple Object Access Protocol)

SOAP es un protocolo para el intercambio de información en un ambiente descentralizado y distribuido. Es el protocolo más utilizado para realizar el intercambio de información en el modelo de web services.

Está basado en XML y potencialmente puede ser utilizado en combinación con una variedad de protocolos de comunicación, siendo el más utilizado HTTP. Por lo tanto se utiliza HTTP para transportar la información, y XML para representar la misma.

El modelo de comunicación de SOAP

El modelo de comunicación de SOAP es muy similar al de HTTP. Un cliente hace un requerimiento (request), el servidor que está escuchando los requerimientos lo atiene y responde (response) brindando la información solicitada o enviando un mensaje de error en caso de que el requerimiento no haya sido válido.

El mensaje SOAP consiste en un elemento envelope SOAP obligatorio, un cabezal SOAP opcional y un cuerpo SOAP obligatorio como un documento XML. El cabezal SOAP es utilizado para definir información acerca del requerimiento, mientras que el cuerpo SOAP contiene el método llamado y los parámetros con los que se llama al mismo.

El mensaje SOAP está compuesto por un envelope (sobre), cuya estructura está formada por los siguientes elementos: header (cabecera) y body (cuerpo).

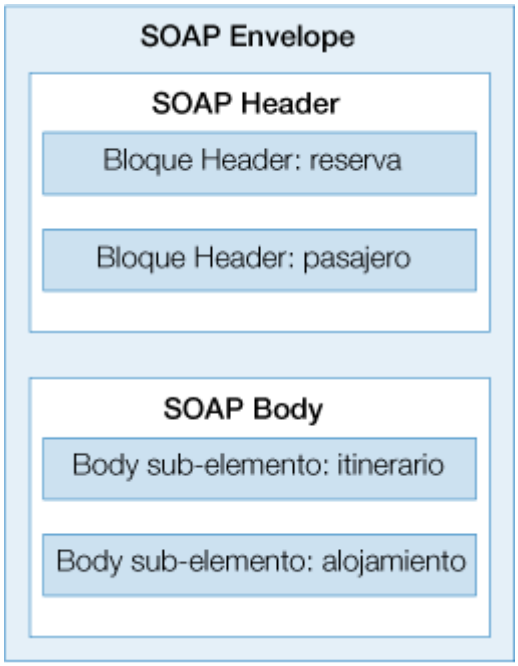


Figura IV.17 Estructura de los mensajes SOAP

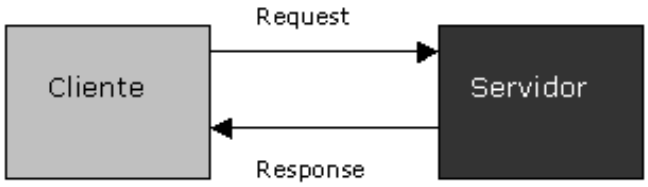


Figura IV.18 El modelo de comunicación

Todo esto es un modelo de mensajes request/response con una forma de describir un conjunto de métodos y pasarle a los mismos parámetros. Esto parece la base del protocolo RPC y de hecho es el uso más común de SOAP. El potencial es entregar esto sobre Internet utilizando HTTP para realizar comunicaciones entre organizaciones permitiendo realizar comunicaciones entre aplicaciones con diferente plataforma, sistema operativo y lenguaje de programación.

A continuación se muestra un mensaje SOAP embebido en un request HTTP:

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
     <m:GetLastTradePrice xmlns:m="Some-URI">
        <symbol>DIS</symbol>
     </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura IV.19 Mensaje SOAP

Este ejemplo invoca al servicio StockQuote llamando al método GetLastTradePrice con el símbolo DIS por parámetro. Este es la respuesta al requerimiento anterior, el cual retorna el precio de la acción solicitada:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Body>
     <m:GetLastTradePriceResponse xmlns:m="Some-URI">
        <Price>34.5</Price>
     </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura IV.20 Respuesta SOAP

WSDL: Web Services Description Language

WSDL es un lenguaje basado en XML que se utiliza para describir un Web Services. Ha sido suministrado por la W3C por estandarización.

Un archivo con formato WSDL provee información de los distintos métodos (operaciones) que el Web Services brinda, muestra cómo accederlos y que formatos deben de tener los mensajes que se envían y se reciben. Es como un contrato entre el proveedor del servicio y el cliente, en el cual el proveedor se compromete a brindar ciertos servicios solo si el cliente envía un requerimiento con determinado formato. Es el documento principal a lo hora de documentar un Web Services, pero puede no ser el único.

En forma resumida podríamos decir que un archivo WSDL describe lo siguiente:

- Mensajes que el servicio espera y mensajes que el servicio responde.
- Protocolos que el servicio soporta.
- A donde mandar los mensajes.

Formato de un archivo WSDL:

A continuación se muestra como es el formato básico de un archivo WSDL. La especificación completa de este lenguaje se puede encontrar en http://www.w3.org/TR/wsdl.html

Un archivo con formato WSDL básicamente contiene los siguientes elementos:

Type: Describe los tipos no estándar usados por los mensajes (Message).

Message: Define los datos que contienen los mensajes pasados de un punto a otro.

PortType: Define una colección de operaciones brindadas por el servicio. Cada operación tiene un mensaje de entrada y uno de salida que se corresponde con algún Message antes definido.

Binding: Describe los protocolos que se utilizan para llevar a cabo la comunicación en un determinado PortType; actualmente los protocolos soportados son SOAP, HTTP GET, HTTP POST y MIME, siendo SOAP el más utilizado.

Port: Define una dirección (URL) para un determinado Binding

Service: Define una colección de Ports.

- **Servicios web en JAVAEE (jax ws)**

El API de Java para servicios web basados en XML es JAXWS es la especificación que define el estándar para servicios web en JavaEE,

¿Qué es JAXWS?

JAX WS (Java Api for XML Web Services) es el centro de una nueva rearquitectura para web services de Sun, la cual incluye JAXB 2.0 y SAAJ 1.3.

JAX WS forma parte del estándar Java EE, el cual reemplaza y amplía al anterior API de accesos a servicios Web (JAX RPC), aunque actualmente ambos están en uso. Una ventaja de JAX WS es que hace uso de anotaciones Java para simplificar el desarrollo.

La especificación JAX WS proporciona soporte para servicios web que utiliza la API JAXB para vincular datos XML en objetos Java. La especificación JAX WS define APIs clientes para acceder a servicios web así como técnicas para implementar puntos finales de servicios web. Los servicios web para la especificación JavaEE describen el despliegue de servicios basados en clientes JAX WS. La especificación EJB y servlet también describe aspectos de ese desarrollo. Esto debe posibilitar el despliegue de aplicaciones basadas en JAX WS utilizando cualquiera de estos modelos de despliegue.

La especificación JAX WS describe el soporte para los manejadores de mensajes que pueden procesar mensajes de solicitud y respuesta. En general, estos manejadores de mensajes se ejecutan en el mismo contenedor y con los mismo privilegios y contextos de ejecución que el JAX WS cliente o en el componente punto final con el que está asociado.

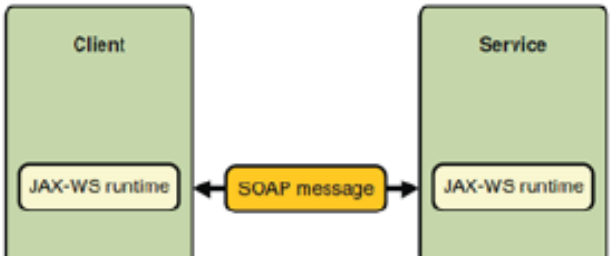


Figura IV.21 API JAX WS

El siguiente cuadro muestra las APIs disponibles en JavaEE y sus paquetes correspondientes.

Tabla IV.3 Paquetes APIs JAX WS

Paquete	API
javax.xml.ws	Núcleo del API JAX WS
javax.xml.soap	API para crear y construir mensajes SOAP
javax.jws	Metadatos para la construcción de Servicios Web

Beneficios JAX WS

1. Simplifica el desarrollo de aplicaciones que exponen servicios WEB
2. Plataforma independiente de JAVA
3. Permite utilizar servicios de distintas tecnologías, no todos deben ejecutarse en plataforma JAVA
4. Usa tecnologías definidas por el consorcio W3C
5. Utiliza HTTP, SOAP y WSDL para describir el servicio.

Definición de servicios web con JAX WS

1. El único requisito es contar con un interfaz y/o una clase de implementación anotado con @WebService

- ✓ En el caso de EJB endpoints , además deben de estar anotados como @Stateless (los servicios web son sin estado)

2. La clase de implementación debe ser pública y no puede ser final ni abstract
3. La clase de implementación debe contar con un constructor vacío
4. La clase de implementación no puede definir un método finalize()
5. Debe garantizarse una implementación sin estado
   - ✓ La clase de implementación no puede guardar info. de estado entre llamadas del cliente
6. Por defecto, para la clase/interface de implementación:
   - ✓ Se generará un elemento WSDL service con el mismo nombre de la clase y el sufijo Service
   - ✓ Se generará un elemento WSDL portType con el nombre de la clase
7. Para cada método público de la clase se generará:
   - ✓ Un elemento WSDL operation con el mismo nombre del método
   - ✓ Dos elementos WSDL message: uno para la petición (con el nombre del método) y otro para la respuesta (añadiendo al nombre del método el sufijo respose)
   - ✓ Los parámetros y valores de retorno deben de ser tipos básicos Java, clases anotadas con JAXB o arrays, Map, List o Collection de los anteriores

Anotaciones JAX WS

Anotaciones que definen el mapeo WSDL (modifican el comportamiento por defecto):

@WebService.

Señala una clase o interfaz como endpoint de un servicio web

- ✓ incluye atributos para modificar el nombre del elemento service, portType, el namespace, etc (name,targetNamespace, serviceName, portName, wsdlLocation, endpointInterface)

@WebMethod.

Permite modificar la definición de las operaciones WSDL (atributo operationName) o excluir métodos de la clase que no se desena exponer comooperaciones del web service (con el atributo exclude=true)

@WebResult

Permite controlar el nombre del elemento message de WSDL que contendrá el valor de retorno (atributo name)

@WebParam.

Permite configurar los elementos parameter de WSDL vinculados a los parámetros de una operación (atributos: name, mode [IN, OU, INOUT], targetNamespace, header, partName)

@OneWay.

Permite indicar que un método no tendrá valor de retorno

Anotaciones que definien el binding SOAP de las operaciones/métodos

@SOAPBinding.

Para un método de la clase endpoint especifica el estilo de codificación de los mensajes (RPC vs. document) y el tipo de codificación de los parámetros a usar (encoded vs.literal).

Atributos: style, use, parameterStyle.

- **@SOAPMessageHandle**

Especifica detalles de la gestión de los mensajes (petición y respuesta)

Atributos: name, className, initParams, roles, heards

Ejemplo de un Servicio Web con JAXWS

```
@WebService(name = “CreditCardValidator”,portName=”ValidatorPort”)
public class CardValidator {
    @WebMethod(operationName = “ValidateCreditCard”)
    @WebResult(name “IsValid”)
    public boolean validate (
        @WebParam(name = “CreditCard”) CreditCard creditCard)
        String lastDigit = creditCard.getNumber().toString(
                creditCard.getNumber().length() – 1,
                creditCard.getNumber().length());
        if (Integer.parseInte(lastDigit) % 2 != 0) {
            return true;
        }
        else{
            return false;
        }
    }
@WebMethod(excluded = true)
    Public void validate( String ccNumber)^
    {
        //Logica de negocios
    }
```

o **Implementación de Servicios Web**

A continuación vamos a crear un Servicio Web utilizando jax ws, utilizando Netbeans 6.9.1

Paso1

En el menú Archivo, seleccione Nuevo proyecto. Debería ver el cuadro de diálogo que se muestra en la Figura IV.7.

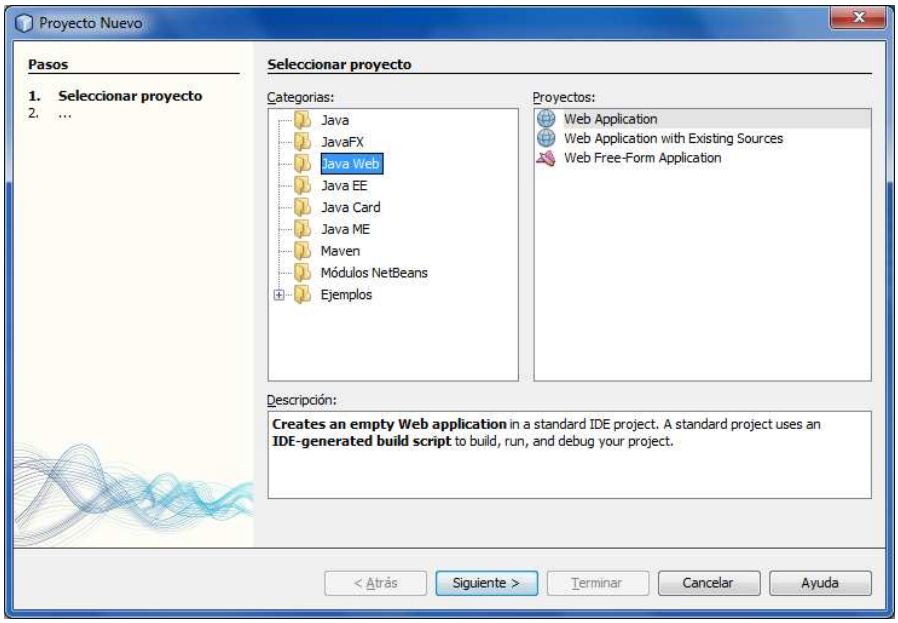


Figura IV.22 IDE NetBeans 6.9 Cuadro de Diálogo Nuevo Proyecto

Paso 2.

En la columna Categorías, seleccione Java Web. En la ventana Proyectos, seleccione Aplicación Web. Haga clic en Siguiente, y el cuadro de diálogo debe ser similar a la Figura IV.8.

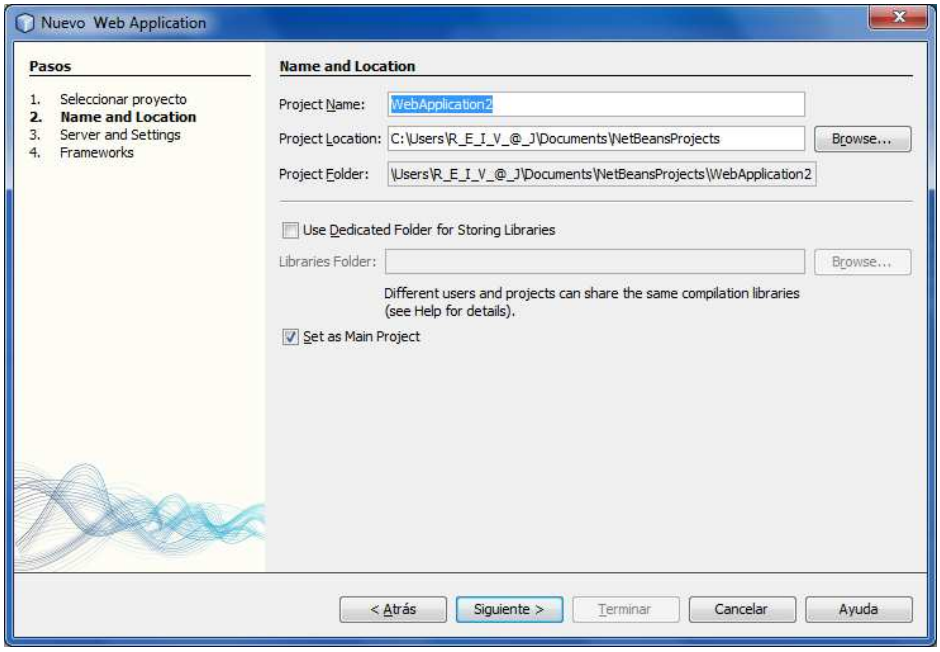


Figura IV.23 Cuadro de diálogo Nueva aplicación WEB

Paso 3.

En el cuadro de dialogo Nombre del proyecto ponga ej. EjemploWS, los campos de la ruta en donde se almacenara su proyecto no los cambie, luego presione siguiente como se muestra en la Figura IV.9.

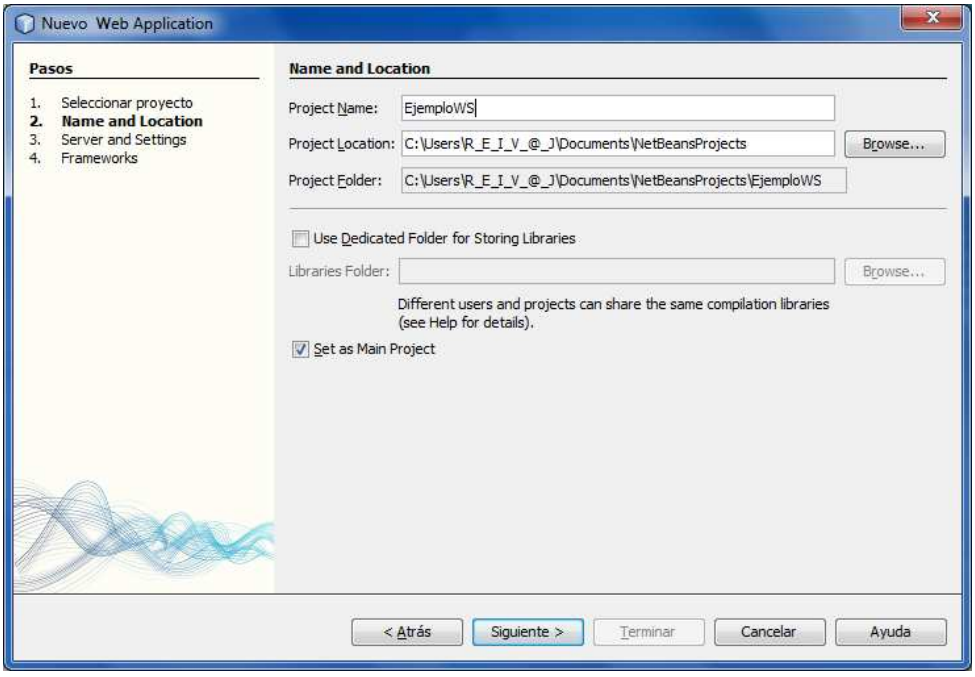


Figura IV.24 Cuadro de diálogo Nombre del Proyecto

Paso 4.

Seleccione el servidor donde se va a alojar su aplicación, en el combobox server. En vuestro caso seleccionar glassfhish que está instalado en forma nativa. Como se muestra en la Figura IV.10.

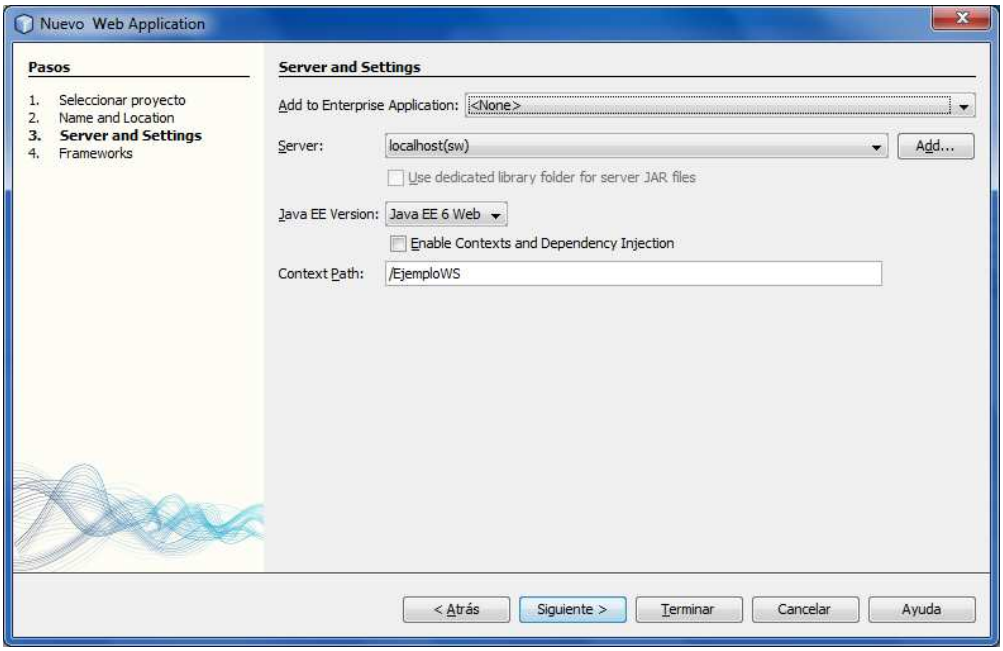


Figura IV.25 Selección del Servidor de prueba para la aplicación

Paso 5.

Una vez creada la aplicación damos clic derecho sobre la misma y seleccionamos. Nuevo >Web Service. Como se muestra en la Figura IV.11.

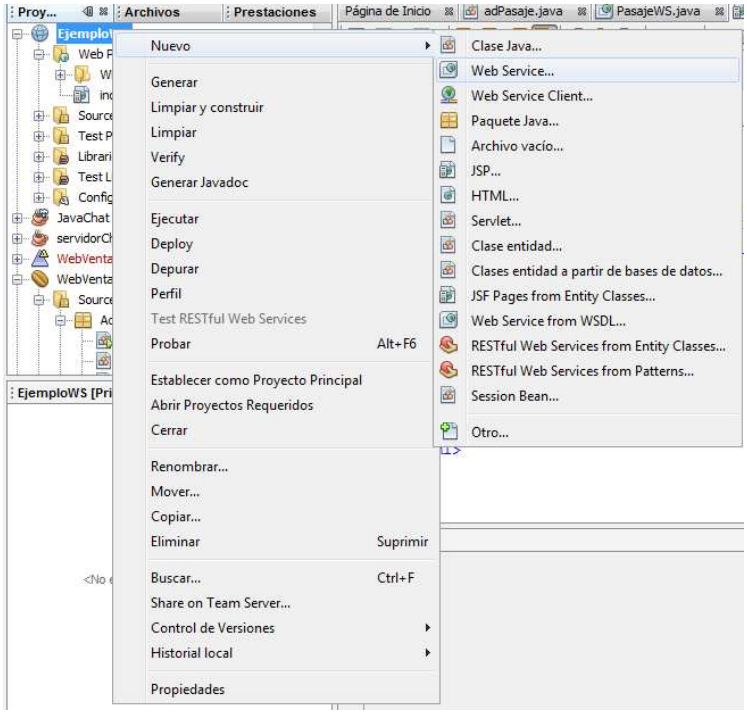


Figura IV.26 Creación de un nuevo WebService

Paso 6.

A continuación veremos un pantalla de dialogo en la cual nos pedirá el nombr de nuestro Web Service y el nombre del paquete en el cual vamos a crear como se muestra en la Figura IV.12.

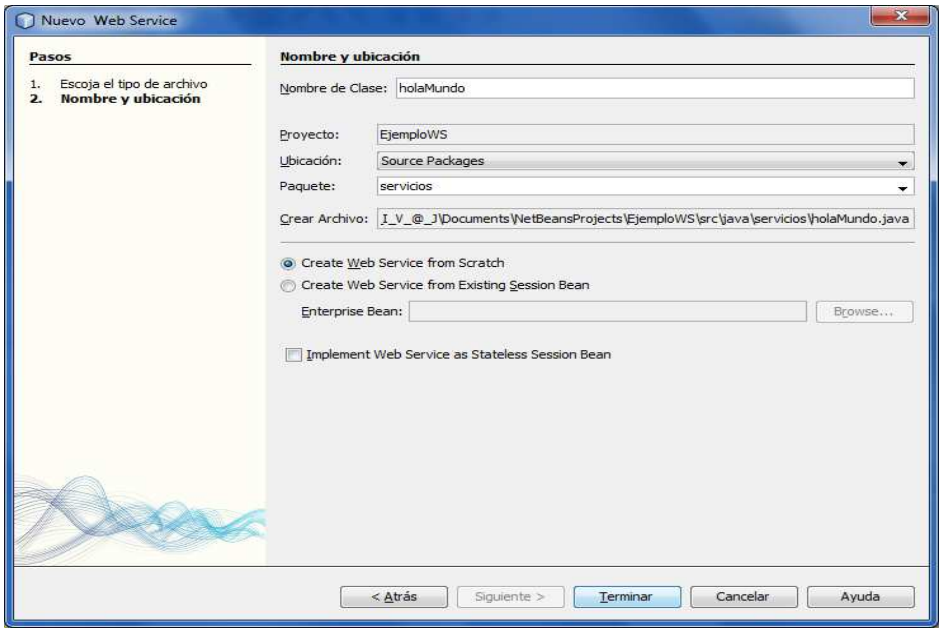


Figura IV.27 Nombre del Servicio Web y paquete a ser creado

Paso 7.

A continuación creamos el código necesario para nuestro servicio web.

```
@WebService()
public class holaMundo {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "mensaje")
    public String mensaje(@WebParam(name = "parametro")
    int parametro) {
        String mensaje;
        if (parametro==0)
        {
         mensaje = "Hola como estan "+parametro;
        }
        else
        {
         mensaje = "Hola como estan "+parametro;
        }
     return mensaje;
    }
}
```

Paso 8.

Pasamos probar si el servicio web como se muestra en la Figura IV.13.

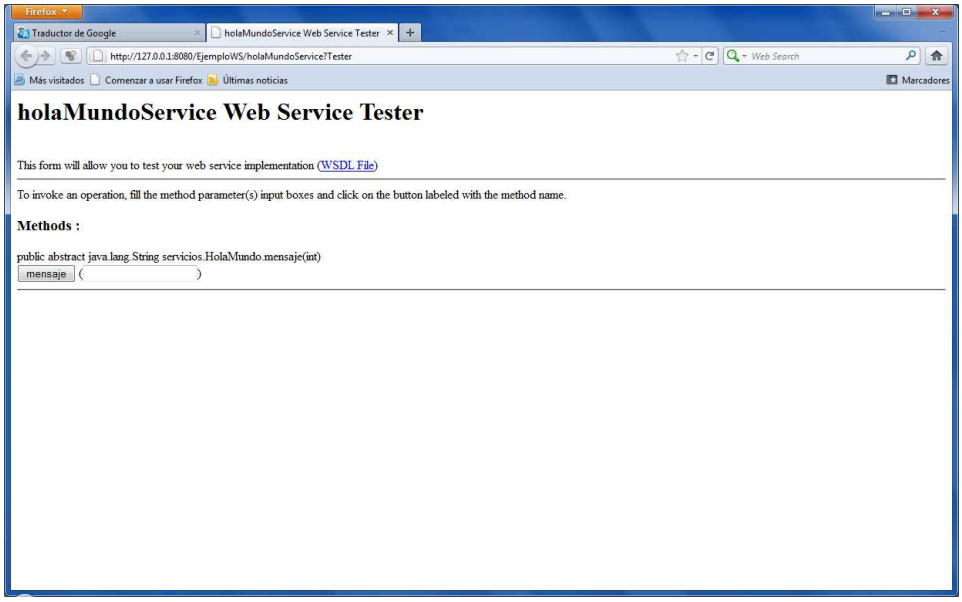


Figura IV.28 Verificación del servicio WEB

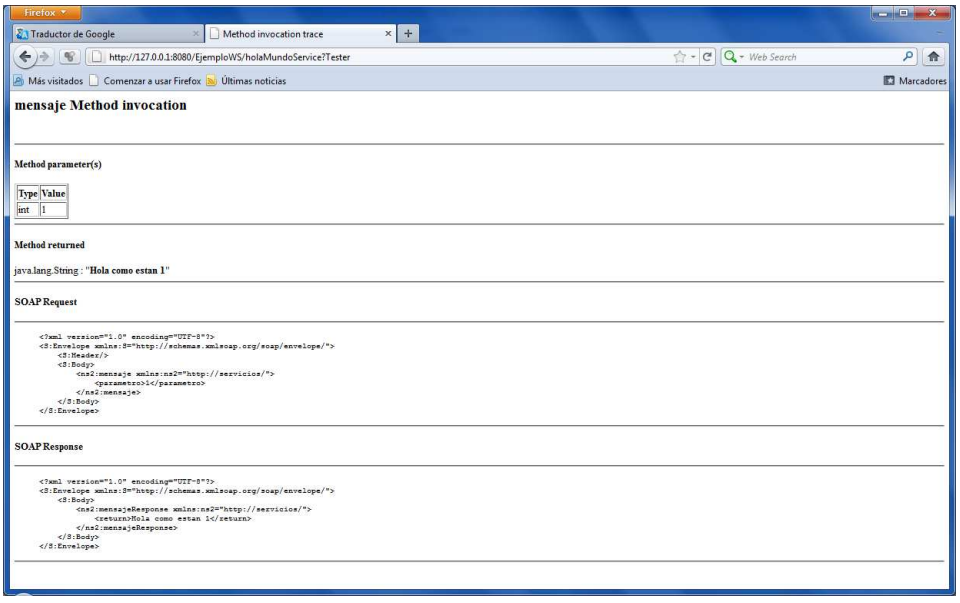


Figura IV.29 Invocación del Servicio WEB

Paso 10.

Podemos ver el wsdl como se muestra en la Figura IV.15

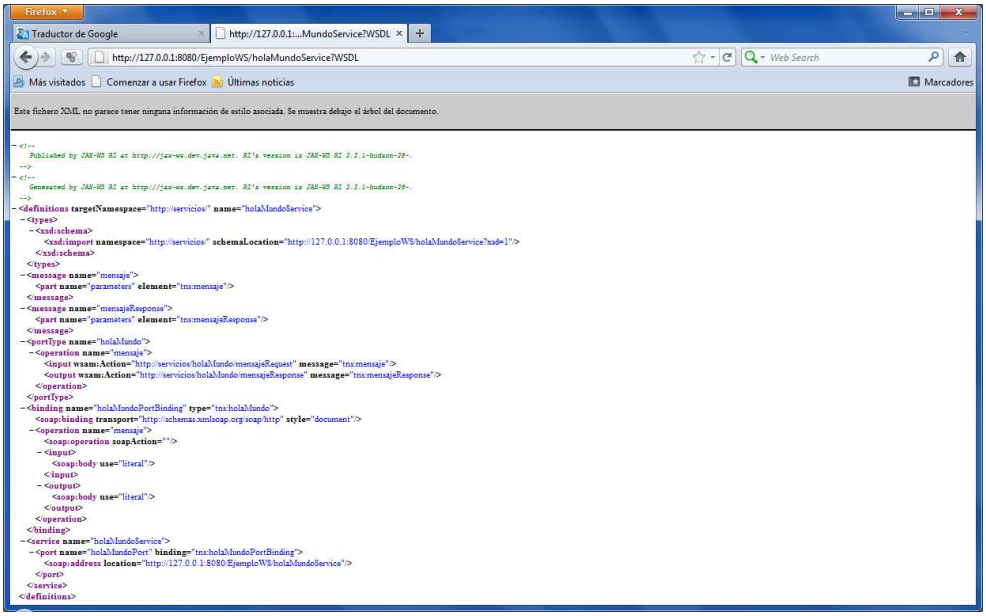


Figura IV.30 Descripción WSDL

CAPITULO V

ESTUDIO COMPARATIVO DE SERVIDORES DE APLICACIONES PARA DESARROLLO DE SOFTWARE CON SOA SOBRE PLATAFORMAS JAVAEE

o INTRODUCCIÓN

Este capítulo se centrará en el establecimiento de los parámetros básicos que permitirán la elección de un servidor de aplicaciones adecuado que permita la futura implementación de una aplicación con arquitectura orientada a servicios, utilizando la plataforma JavaEE.

Estos parámetros se establecen en base al estudio de los temas precedentes como son:

- ✓ Definición de la Arquitectura Orientada a Servicios y las características necesarias para la utilización de este paradigma en la implementación de una solución de tipo empresarial.
- ✓ Definición de la Arquitectura JavaEE y los servidores de aplicaciones existentes en el medio que implementen esta arquitectura referencial posibilitando la elaboración de aplicaciones SOA.
- ✓ Definición de los lineamientos propuestos para la implementación de SOA con Servicios Web y la referencia del API JAX WS de JavaEE.

Se ha establecido para el desarrollo de la presente investigación, aunque no es la única forma de elaborar soluciones orientadas a servicios, la implementación mediante

Servicios WEB, tomando en cuenta que este estándar se encuentra debidamente soportado en múltiples plataformas, y permite el acceso a los Servicios expuestos a través del protocolo HTTP, el cual es soportado no solo por computadores, sino también por dispositivos móviles y cualquier otra implementación futura que tenga acceso a Internet.

Logrando de esta manera cumplir con otro de los principios imperantes de SOA, el cual es brindar el acceso a los servicios de aplicación sin importar la plataforma y el lenguaje de implementación del lado del cliente. Cabe indicarse que hay servicios que consumen información de otros, constituyéndose los segundos en clientes, que posiblemente no se encuentren implementados en el mismo leguaje y plataforma que los primeros.

- **SERVIDORES DE APLICACIONES JAVA PARA SOLUCIONES CON SOA**

Se necesita entonces tomar la mejor decisión sobre qué servidor de Aplicaciones JavaEE se ajusta a las capacidades exigidas para el despliegue de una aplicación que permita el acceso a la información desde distintas plataformas, y que cada implementación adicional no signifique una reingeniería del proceso completo, que haga incurrir en gastos significativos a la organización o empresa.

Existe una gran cantidad de opciones que implementan arquitectura Java EE, y tomando como referencia la investigación realizada por Jonathan Campbell en colaboración con ORACLE, en su blog se puede distinguir a cuatro servidores de aplicaciones que se utilizan mayoritariamente en el mundo empresarial. Todos tienen sus versiones libres mantenidas por Comunidades de Desarrolladores en Internet, y adicionalmente las versiones de paga, que se diferencian de las anteriores porque implementan mecanismos más robustos en lo que tiene que ver a Seguridad.

Los servidores expuestos entonces son: Oracle open source GlassFish v3, RedHat open source JBoss v5, IBM WebSphere v9 y Oracle WebLogic v10.

Todos estos servidores se someterán al estudio previamente configurados en equipos virtuales independientes con el Sistema Operativo Linux CentOS v5.5, con el mismo ambiente operativo tanto en hardware como en software.

- **ESTABLECIMIENTO DE LOS PARÁMETROS PARA LA ELECCIÓN DE UN SERVIDOR DE APLICACIONES PARA SOA.**

Las características que se exponen aquí son Estándares de la industria propuestos por varios autores y expertos en el desarrollo de aplicaciones SOA para la plataforma JavaEE:

- ✓ Cumplimiento del estándar Java EE 5.
- ✓ Posibilidad de despliegue JSP y Servlets.
- ✓ Posibilidad de despliegue EJB.
- ✓ Posibilidad de despliegue Java Server Faces.
- ✓ Posibilidad de despliegue ADF.
- ✓ Posibilidad de personalización mediante plugins.
- ✓ Soporte mediante Motor de Reglas de Negocio.
- ✓ Soporte del Framework Hibernate.
- ✓ Soporte de escalabilidad mediante Clustering.
- ✓ Soporte del API JAX WS / JAX B.
- ✓ Integración con el IDE Eclipse.
- ✓ Integración con el IDE NetBEANS.
- ✓ Integración con el IDE JDEveloper.

- **ESTUDIO COMPARATIVO**

Tabla V.4 Parámetros para la elección de un servidor de aplicaciones para SOA

	SERVIDORES DE APLICACIÓN			
Parámetro de Evaluación	**GlassFish**	**JBoss**	**WAS**	**WebLogic**
Cumplimiento del estándar Java EE 5.	Sí	Parcial	Si	Si
Posibilidad de despliegue JSP y Servlets.	Sí	Sí	Sí	Sí
Posibilidad de despliegue EJB.	Sí	Sí	Sí	Sí
Posibilidad de despliegue Java Server Faces.	Sí	Sí	Sí	Sí
Posibilidad de despliegue ADF.	Si	Disponible	Disponible	Si
Posibilidad de personalización mediante plugins.	Sí	Sí	Sí	Sí
Soporte mediante Motor de Reglas de Negocio.	Sí	Disponible	Disponible	Disponible
Soporte del Framework Hibernate.	Si	Sí	Disponible	Disponible
Soporte de escalabilidad mediante Clustering.	Si	Si	Si	Si
Soporte del API JAX WS / JAX B.	Sí	Disponible	Sí	Sí

Integración con el IDE Eclipse.	Disponible	Disponible	Disponible	Disponible
Integración con el IDE NetBEANS.	Sí	Disponible	Disponible	Disponible
Integración con el IDE JDeveloper.	Sí	Disponible	No	Sí
Integración con el IDE JBoss Developer Studio	No	Si	No	No
Integración con el IDE IBM Rational Application Developer Studio	No	No	Si	No

Basada en el estudio de Jonathan Campbell, Desarrollador de Software e Ingeniero de Redes, poseedor de quince certificaciones de IT, especializado en Java, C++, C#, administración de proyectos de IT, Ingeniería de redes en plataformas AIX, Linux y Microsoft.

El estudio de referencia inicial elaborado por Jonathan Campbell en febrero del año 2008 ha sido actualizado, en razón a que las versiones de los servidores de aplicaciones sometidas al estudio comparativo han evolucionado sus implementaciones para estar acorde a los estándares de Referencia JavaEE. Es así que las versiones uilizadas para los cuatro servidores son:

- ✓ GlassFish V3
- ✓ JBoss V6
- ✓ WebSphere V7
- ✓ WebLogic V10

Tabla V.5 Tabla de valoración de Parámetros para la elección del SA para SOA

Detalle	Valor
SI	4
Disponible	3
Parcial	2
No	1

Tabla V.6 Parámetros para la elección de un SA para SOA (Ponderaciones)

	SERVIDORES DE APLICACIÓN			
Parámetro de Evaluación	**GlassFish**	**JBoss**	**WAS**	**WebLogic**
Cumplimiento del estándar Java EE 5.	4	2	4	4
Posibilidad de despliegue JSP y Servlets.	4	4	4	4
Posibilidad de despliegue EJB.	4	4	4	4
Posibilidad de despliegue Java Server Faces.	4	4	4	4
Posibilidad de despliegue ADF.	4	3	3	4
Posibilidad de personalización mediante plugins.	4	4	4	4
Soporte mediante Motor de Reglas de Negocio.	4	3	3	3
Soporte del Framework Hibernate.	4	4	3	3
Soporte de escalabilidad mediante Clustering.	4	4	4	4
Soporte del API JAX WS / JAX B.	4	3	4	4

Integración con el IDE Eclipse.	3	3	3	3
Integración con el IDE NetBEANS.	4	3	3	3
Integración con el IDE JDeveloper.	4	3	1	4
Integración con el IDE JBoss Developer Studio	1	4	1	1
Integración con el IDE IBM Rational Application Developer Studio	1	1	4	1
Totales	**53**	**49**	**49**	**50**

Explicación.

De la tabla anterior si todos los servidores cumplieran con todos los parámetros de evaluación se lograría la suma de 60 lo que representaría el 100%;

De tal manera para la representación se empleara una regla de tres simple, de la siguiente manera.

60	100%	
53	X	= (53*100)/60 = 0,88333 = 88.33%

Obteniendo los siguientes resultados.

Tabla V.7 Porcentajes de Análisis comparativo de SA para SOA

GlassFish	JBoss	WAS	WebLogic
88,33%	81,67%	81,67%	83,33%

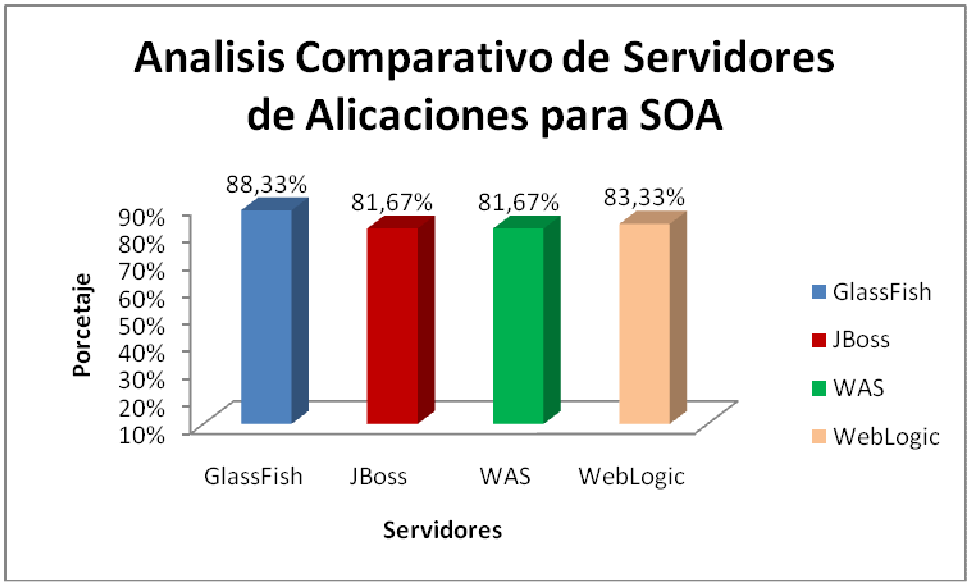


Figura V.31 Análisis Estadístico de Servidores de Aplicaciones para SOA

Resultados Obtenidos.

GlassFish es el servidor de aplicaciones que obtiene el mejor resultado con un 88.33%, debido a sus características, integración con herramientas, facilidad de acoplamiento, Con un 83.33% WebLogic está ubicado en la segunda posición y en la misma posición JBoss y WAS con un porcentaje de 81.67%.

GlassFish es un servidor de código abierto, de fácil instalación, posee soporte completo con Java EE, integración total con la mayoría de IDES, incluye las nuevas librerías de Web Services (JAX WS) y es la base de las nuevas plataformas SOA en JAVA. Los demás servidores poseen deficiencias en algunas de las características mencionadas en la tabla 7.

Hipótesis

“La utilización de un servidor de aplicaciones adecuado, permitirá la interoperabilidad entre la Arquitectura Orientada a Servicios y la plataforma JavaEE, reduciendo los tiempos de respuesta de acceso a la información.”

En base al estudio comparativo de los servidores de aplicaciones propuestos y su grado de cumplimiento en los parámetros definidos, después de haber establecido estos mediante el estudio de la arquitectura orientada a servicios, se ha procedido a elegir el servidor de aplicaciones adecuado.

Entonces el objetivo primordial de la presente investigación es demostrar que el servidor de aplicaciones elegido una vez realizado el estudio comparativo, asegurará la interoperabilidad y brindará el menor tiempo de respuesta de acceso a la información de las aplicaciones cliente.

INTEROPERABILIDAD. Condición mediante la cual los sistemas heterogéneos pueden intercambiar procesos y datos. Un sistema implementado con SOA debe ser interoperable, esto se logra brindando acceso completo a la información disponible. Y la plataforma Java hace posible el desarrollo de aplicaciones para tres ambientes de despliegue como son:

- ✓ JavaEE. aplicaciones distribuidas multicapa sobre web.
- ✓ JavaSE. aplicaciones de escritorio y applets
- ✓ JavaME. aplicaciones para dispositivos móviles. Es una versión simplificada del JavaSE mas APIs específicas.

Si se hace referencia a las características de la Arquitectura SOA, esta propone que la información se exponga mediante servicios con las siguientes consideraciones:

- ✓ Servicios Descubribles
- ✓ Servicios Modulares
- ✓ Servicios Interoperables
- ✓ Servicios de Acoplamiento bajo
- ✓ Servicios con interfaces de granularidad gruesa
- ✓ Servicios Transparentes de localización
- ✓ Servicios Auto recuperables.

Entonces para demostrar la interoperabilidad de la arquitectura propuesta, definimos los siguientes indicadores:

- ✓ Variación del Tipo de Dato.
- ✓ Utiliza el mismo protocolo para el acceso (HTTP)
- ✓ Utiliza el mismo protocolo para su descripción (WSDL)
- ✓ Utiliza UDDI para su descubrimiento.
- ✓ Número de líneas de código.

- ✓ Variación de Código para el acceso a la información.

Medición del porcentaje de interoperabilidad entre JavaEE e indicadores de SOA:

Tabla V.8 Servidor GlassFish servicio web Tabla Código

No.	Indicador	JavaEE	JavaSE	JavaME
1	Variación del Tipo de Dato.	No	No	Si
2	Utiliza el mismo protocolo para el acceso (HTTP)	Si	si	Si
3	Utiliza el mismo protocolo para su descripción (WSDL)	Si	Si	Si
4	Utiliza UDDI para su descubrimiento.	Si	Si	Si
5	Variación de Código para el acceso a la información.	No	No	Si

Tabla V.9 Servicio web Tabla Código GlassFish (Ponderaciones)

Servicio Web Tabla Código		
Indicador	No	Si
1	20%	0%
2	0%	20%
3	0%	20%
4	0%	20%

5	20%	0%

JavaEE: 100%

JavaSE: 100%

JavaME: 60%

Promedio: 86.66%

Tabla V.10 Servidor JBoss Servicio web Tabla Codigo

No.	Indicador	JavaEE	JavaSE	JavaME
1	Variación del Tipo de Dato.	No	No	Si
2	Utiliza el mismo protocolo para el acceso (HTTP)	Si	si	Si
3	Utiliza el mismo protocolo para su descripción (WSDL)	Si	Si	Si
4	Utiliza UDDI para su descubrimiento.	Si	Si	Si
5	Variación de Código para el acceso a la información.	si	si	Si

Tabla V.11 Servicio Web Tabla Código JBoss(Ponderaciones)

Servicio Web Tabla Código		
Indicador	No	Si
1	20%	0%
2	0%	20%
3	0%	20%
4	0%	20%
5	20%	0%

JavaEE: 80%

JavaSE: 80%

JavaME: 60%

Promedio: 73,33%

Tabla V.12 Servidor WebSphere Servicio Web Tabla Código

No.	Indicador	JavaEE	JavaSE	JavaME
1	Variación del Tipo de Dato.	No	No	Si
2	Utiliza el mismo protocolo para el acceso (HTTP)	Si	si	Si
3	Utiliza el mismo protocolo para su descripción (WSDL)	Si	Si	Si
4	Utiliza UDDI para su descubrimiento.	Si	Si	Si
5	Variación de Código para el acceso a la información.	No	si	Si

Tabla V.13 Servicio Web Tabla Código WebSphere (Ponderaciones)

Servicio Web Tabla Código		
Indicador	No	Si
1	20%	0%
2	0%	20%
3	0%	20%
4	0%	20%
5	20%	0%

JavaEE: 100%

JavaSE: 80%

JavaME: 60%

Promedio: 80%

Tabla V.14 Servidor WebLogic Servicio Web Tabla Código

No.	Indicador	JavaEE	JavaSE	JavaME
1	Variación del Tipo de Dato.	No	No	Si
2	Utiliza el mismo protocolo para el acceso (HTTP)	Si	si	Si
3	Utiliza el mismo protocolo para su descripción (WSDL)	Si	Si	Si
4	Utiliza UDDI para su descubrimiento.	Si	Si	Si
5	Variación de Código para el acceso a la información.	No	No	Si

Tabla V.15 Servicio Web Tabla Codigo WebLogic (Ponderaciones)

Servicio Web Tabla Código		
Indicador	No	Si
1	20%	0%
2	0%	20%
3	0%	20%
4	0%	20%
5	20%	0%

JavaEE: 100%

JavaSE: 100%

JavaME: 60%

Promedio: 86.66%

Interpretación de Resultados

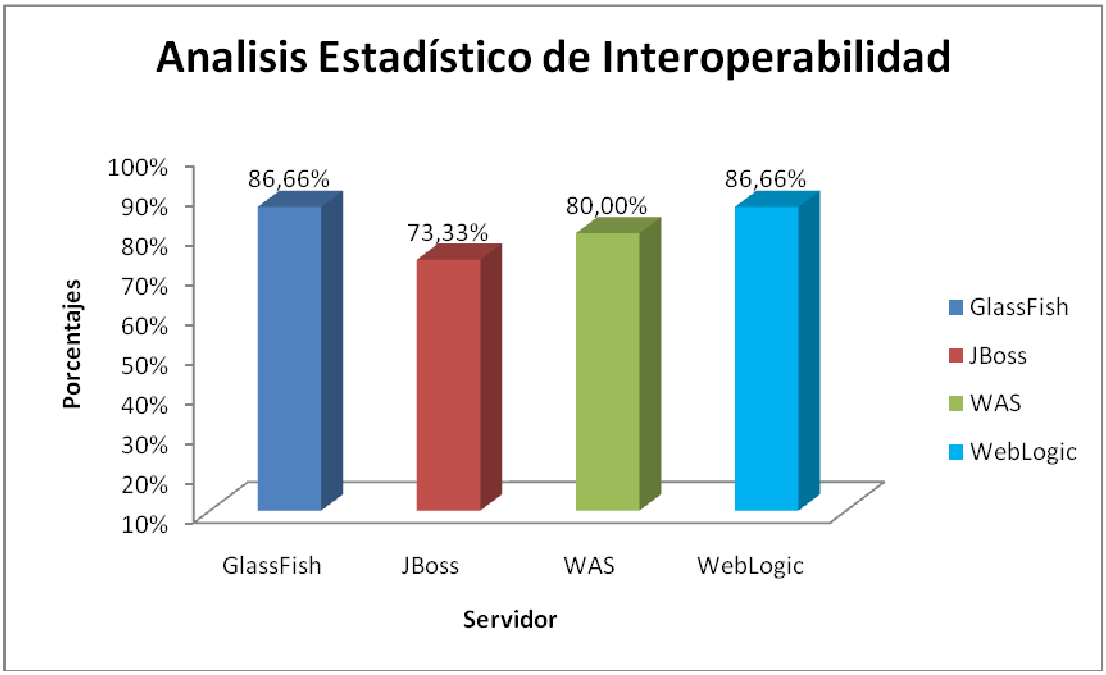


Figura V.32 Análisis Estadístico de Interoperabilidad de SA

Resultados Obtenidos.

GlassFish y WebLogic son los servidores que obtienen el mejor resultado con un 86.66%, debido a interoperabilidad con la platoforma Java EE (Java SE, Java EE, JavaME) ,debido a que presenta una integración con los servicios web. Con un 80.00% WAS está ubicado en la segunda posición y en la tercera posición JBoss con un porcentaje de 73.33%.

Análisis del Tiempo de Respuesta.

Para establecer el tiempo de respuesta que presentan los servidores de aplicaciones propuestos. Se utilizará una técnica denominada Benchmarking. Esta técnica permite determinar el mejor desempeño de varios entes sometidos a comparación, por medio del establecimiento de los puntos de referencia que se obtienen al poner a prueba sus funcionalidades.

Entonces para obtener datos acerca del rendimiento de los servidores de aplicaciones, se va a testear el tiempo que estos se demoran en atender a sesiones de usuario iniciadas vía HTTP.

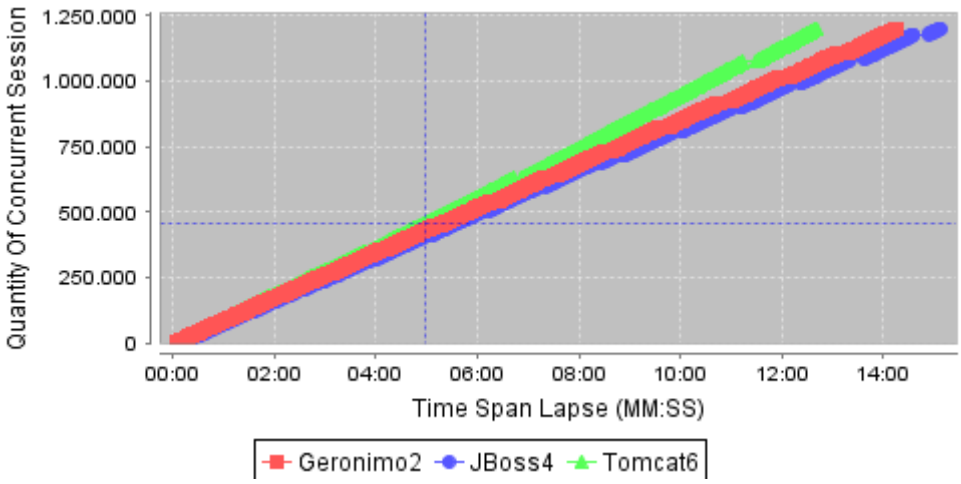


Figura V.33 Sistema de BenchMarking

Procedimiento de Benchmarking

En el servidor se desplegará una aplicación web (archivo .war) que contiene una página JSP y un SERVLET compilado con los cuales se podrá observar, en base al número de sesiones que se establezca como punto de referencia, qué tan rápido puede el servidor conectar a los usuarios.

Estas sesiones se crearán de manera concurrente desde un cliente remoto. Este cliente poseerá un aplicativo elaborado en JAVA que provocará el estrés deseado en el servidor para simular un ambiente real de despliegue.

Las aplicaciones JAVA del mundo real tienen un nivel muy grande, y manejan recursos a gran escala, pero mediante este pequeño test se podrá obtener información importante a cerca de la fiabilidad, escalabilidad y velocidad de respuesta de los servidores.

El escenario del BenchMarking

A continuación se describen los componentes del escenario:

- ✓ El equipo disponible para este efecto posee un procesador Core Duo de 32Bits y 2.4GHz de velocidad.
- ✓ La memoria física del sistema es de 3.5GB.
- ✓ El sistema operativo es Linux CentOS 5.4 de 32bits para servidor.
- ✓ La plataforma de virtualización es VMWare Workstation 7.01 para Linux en su versión libre, misma que carece de ciertas características funcionales que no afectaron al presente estudio.
- ✓ Se crearon 4 maquinas virtuales sobre el mismo servidor. Pero durante el proceso de prueba solo se mantenía uno en ejecución.
- ✓ Cada servidor virtual de prueba se configuró con similares características:
    - o Un procesador Dual Core 2.2GHz
    - o Memoria Física de 2GB
    - o 15GB de Espacio en Disco.
    - o Sistema Operativo Linux CentOS 5.4
    - o Entorno de desarrollo OpenJDK 1.6

Con el entorno de servidor completado ya solo resta efectuar la instalación y configuración de los parámetros por separado para cada servidor de aplicaciones como son GlassFish, JBoss, WebSphere y WebLogic. Los detalles de instalación de cada uno de estos se encuentra en el apartado de anexos del presente documento.

Herramientas de BenchMarking

El kit de BenchMarking utilizado en la presente investigación consta de un conjunto de clases, aplicaciones java y secuencias de comandos con las siguientes funcionalidades:

Tabla V.16 Herramientas de BenchMarking

Nombre de Archivo	Tipo	Función
HttpTest_Compile.bat	Procesamiento por lotes	Permite compilar la clase HttpTestB.java cuando han sido editadas por medio de un editor de texto simple.
HttpTest_go.bat	Procesamiento por lotes	Permite la ejecución de los procedimientos necesarios para efectuar el estrés en el servidor. En este archivo se puede editar el numero de sesiones concurrentes a ser evaluadas y la dirección ip del servidor de prueba.
HttpTestB.java	Clase Java	Contiene los métodos necesarios que en base al número se sesiones concurrentes toma el tiempo que el servidor se tarda en efectuar la conexión del cliente. Esta información se refleja en el archivo de texto J2EE_BenchData.
S2.java	Servlet	Permite la obtención del ID de la sesión creada en el servidor.

Set.jsp	Java Server Page	Pagina que contiene la secuencia de comandos que permite crear la sesión en el servidor.
Index.html	Archivo HTML	Contiene el link de acceso a Set.jsp para visualizarlo de forma manual en el servidor cuando se ha desplegado la aplicación.

Los archivos S2.java, Set.jsp e index.html necesitan ser editados y agregados a un nuevo proyecto en cualquier IDE con soporte JAVAEE, en este caso se utilizó NetBeans 6.9 para efectuar la depuración y despliegue de la aplicación Sess.war en los diferentes servidores de Aplicaciones.

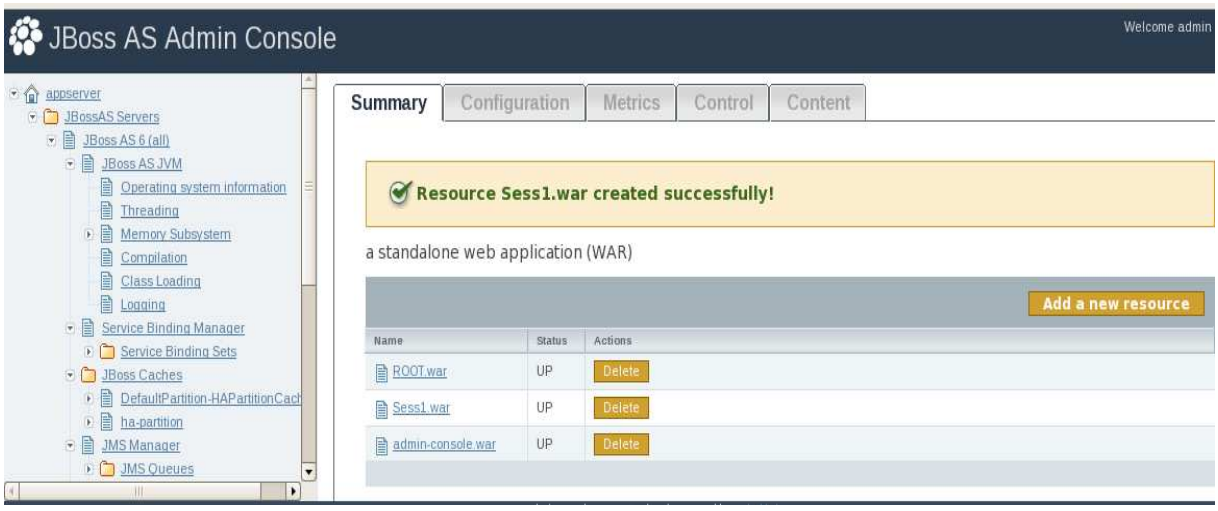


Figura V.34 Despliegue de Sess1.war

Todos los archivos se encuentran dentro de un mismo directorio para posibilitar su ubicación de contexto sin utilizar rutas de acceso completas a los ejecutables.

Procedimiento de BenchMarking

1. Una vez desplegada la aplicación en el Servidor se necesita verificar la url de acceso esté de acuerdo con el formato establecido en el sistema de benchmarking, en este caso:

http://direcciónIP:puerto/Sess1/Set.jsp

http://direcciónIP:puerto/Sess1/S2

2. Entonces se edita el archivo HttpTest_go.bat para indicar el número de sesiones de prueba y la dirección IP del servidor:

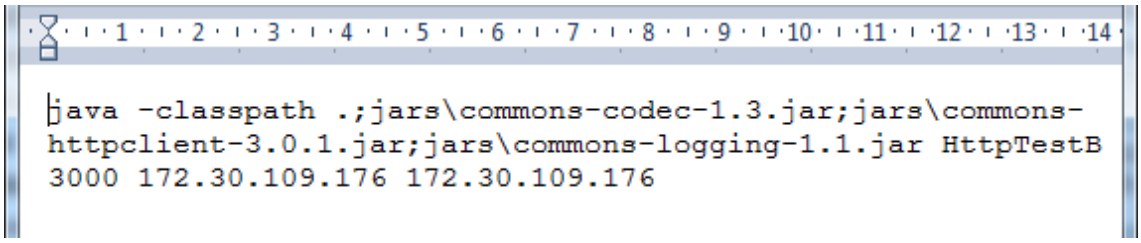


Figura V.35 HttpTest_go.bat

Entonces hemos establecido que el número de sesiones para la prueba en los cuatro servidores va a ser de 3000 sesiones concurrentes. Para la prueba en los otros servidores únicamente se editó la dirección IP.

3. Se procedió a ejecutar el archivo HttpTest_go.bat, para la obtención de los Datos referentes al tiempo de respuesta de conexión de clientes para cada servidor:

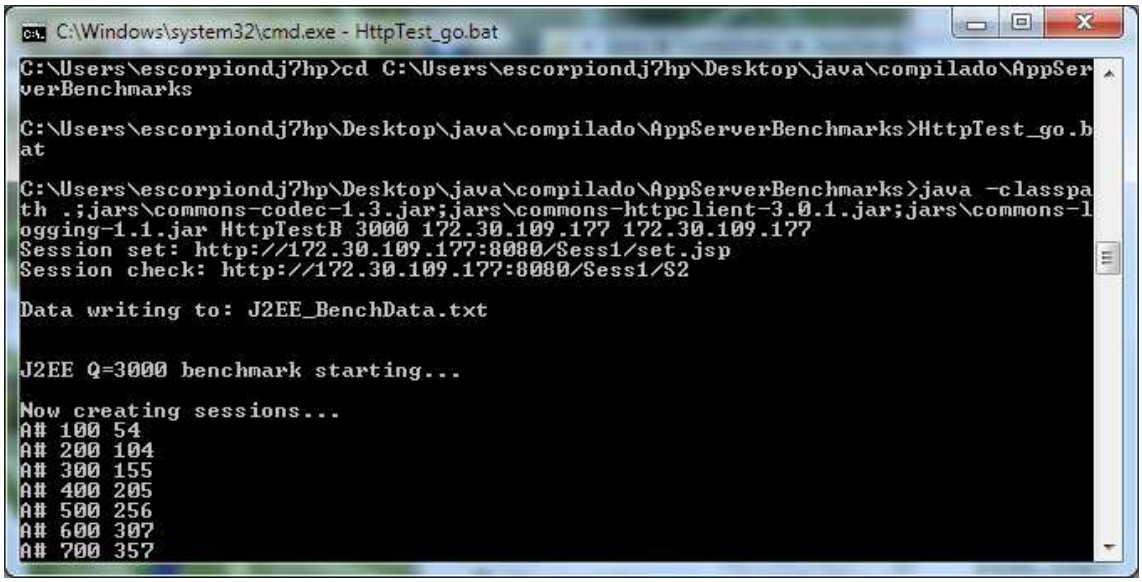


Figura V.36 Ejecución de HttpTest_go.bat

El procedimiento entonces empieza a tomar los datos y escribirlos en segundo plano en el archivo de texto J2EE_BenchData.txt

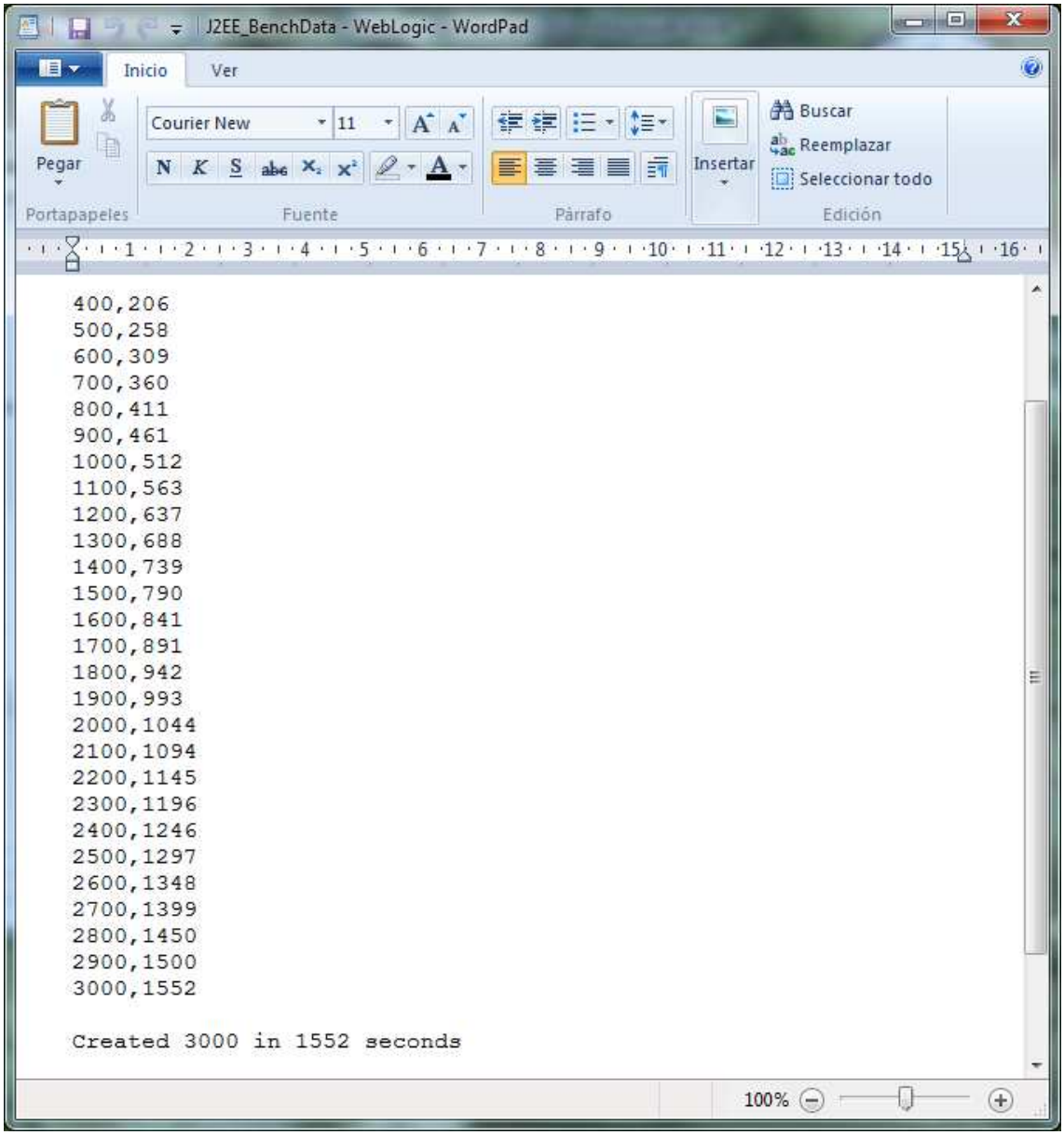


Figura V.37 Resultado de la Ejecución escrito en J2EE_BenchData.txt

4. Finalmente solo resta guardar los datos en archivos identificando a cada servidor para su posterior análisis:

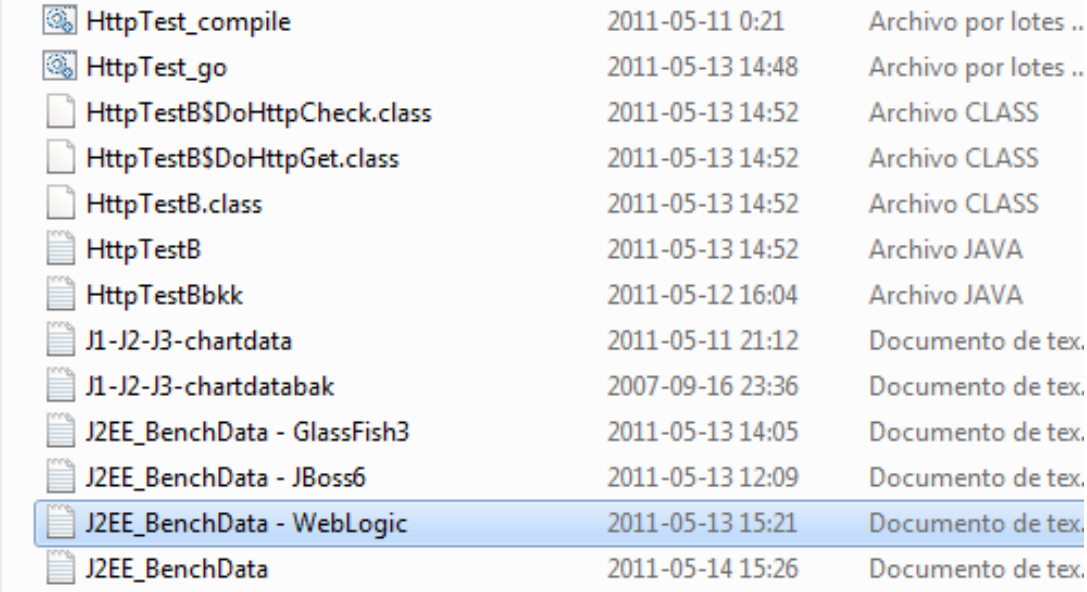


Figura V.38 Datos de los 4 servidores sometidos a prueba

Los datos que se obtuvieron en el proceso de benchmarking se organizaron en la siguiente tabla:

Tabla V.17 Resultados de las peticiones de cada Servidor

Peticiones	Jboss	WebLogic	GlassFish	WebSphere
100	54	53	53	53
200	104	104	104	106
300	155	155	154	157
400	205	206	205	208
500	256	258	256	259
600	307	309	306	310
700	357	360	357	361
800	408	411	408	413
900	458	461	458	463
1000	509	512	509	514
1100	559	563	559	565

1200	610	637	610	639
1300	660	688	660	700
1400	710	739	711	741
1500	761	790	761	792
1600	812	841	812	843
1700	862	891	862	893
1800	913	942	914	946
1900	963	993	964	996
2000	1014	1044	1015	1047
2100	1064	1094	1065	1099
2200	1115	1145	1116	1148
2300	1165	1196	1166	1197
2400	1215	1246	1217	1248
2500	1266	1297	1267	1299
2600	1316	1348	1318	1350
2700	1367	1399	1368	1399
2800	1417	1450	1418	1452
2900	1468	1500	1469	1501
3000	1521	1552	1519	1560

Se analiza la información obtenida sobre los tiempos de respuesta de los servidores al efectuar 3000 conexiones concurrentes:

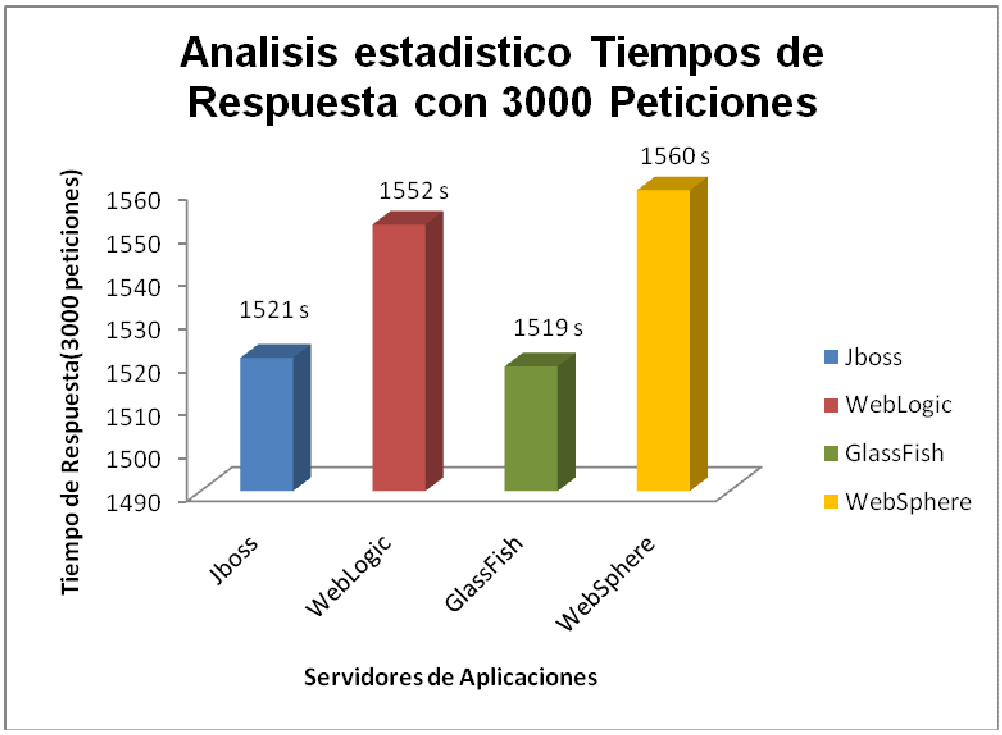


Figura V.39 Análisis estadístico Tiempos de Respuesta con 3000 peticiones

En el grafico se representa el tiempo en segundos que cada servidor se ha tardado en efectuar la conexión de 3000 peticiones concurrentes de clientes, simuladas en este caso por el sistema de BenchMarking.

Situamos en primer lugar al Servidor GlassFish de Oracle con un tiempo de 1519 segundos, el menor tiempo de los servidores sometidos a prueba. En segundo lugar tenemos al servidor JBoss con 1521 segundos, en tercer lugar a WebLogic y en último lugar a WebSphere de IBM.

- **ELECCIÓN DEL SERVIDOR DE APLICACIONES ADECUADO.**

El presente trabajo pretende demostrar que por medio del estudio comparativo de los servidores de aplicaciones JAVAEE, en base al grado de cumplimiento de los parámetros establecidos por el estándar para el desarrollo de aplicaciones SOA, se ha obtenido la mejor opción.

Esta opción al ser sometida al estudio de Interoperabilidad y obtención de los Tiempos de respuesta, ha resultado también favorable con el mayor porcentaje de Interoperabilidad y el menor tiempo de respuesta en el acceso.

Entonces para consolidar los resultados obtenidos se presenta el siguiente cuadro:

Tabla V.18 Consolidado de Valores obtenidos durante el presente estudio.

Servidor de Aplicaciones	Estudio Comparativo (%)	Interoperabilidad (%)	Tiempo de Respuesta 3000 peticiones(segundos)
GlassFish	88.33%	86.66%	1519s
JBoss	81.67%	73.33%	1521s
WebSphere	81.67%	80%	1560s
WebLogic	83.33%	86.66%	1552s

Se demuestra entonces que una vez realizado el estudio comparativo, en el cual el servidor GlassFish obtiene el mayor porcentaje de cumplimiento de los estándares junto con WebLogic, y posteriormente sometidos a las pruebas de interoperabilidad y rendimiento, GlassFish obtiene el menor tiempo de respuesta de acceso a la información.

Entonces se comprueba de manera favorable la hipótesis planteada, ya que GlassFish es el servidor de aplicaciones adecuado que asegurará la interoperabilidad y brindará el menor tiempo de respuesta de acceso a la información.

CONCLUSIONES

- ✓ La Arquitectura Orientada a servicios propone un marco de desarrollo para implementar aplicaciones con bajo acoplamiento que comparten información accesible entre plataformas heterogéneas.
- ✓ El empleo de SOA en empresas asegura bajos costes de mantenimiento en Aplicaciones Relativamente grandes, debido a la exigencia de bajo acoplamiento entre módulos de componentes.
- ✓ Las tecnologías de servidor se refieren a las plataformas y el lenguaje de programación empleado para el desarrollo de aplicaciones, mientras que los servidores de aplicaciones se constituyen en Software que ofrece servicios de acceso a información de los clientes.
- ✓ Las ventajas de las aplicaciones SOA hacen que la empresa pueda crecer progresivamente, sin la necesidad de emprender en una reingeniería total de los procesos y con el menor costo posible.
- ✓ Los componentes de la arquitectura de un servidor JAVAEE para el soporte de aplicaciones SOA deben incluir implementaciones para la publicación de servicios WEB, transportes de mensajería y métodos de autenticación de clientes.
- ✓ La plataforma JAVAEE ofrece una variedad de Servidores de Aplicaciones tanto libres como propietarios con abundante documentación y soporte, siempre cumpliendo con los estándares de carácter empresarial que permiten ajustarse a los lineamientos del desarrollo de soluciones SOA.
- ✓ El servidor de Aplicaciones GlassFish se constituye en la mejor opción para SOA, ya que es la implementación directa de la empresa elaboradora de los estándares SOA y JavaEE.

- ✓ GlassFish cumple con los parámetros establecidos en la presente investigación tanto en Implementación del Estándar, Porcentaje de Interoperabilidad y el mejor tiempo de respuesta.
- ✓ WebLogic ofrece la mejor calidad de Soporte en Línea, flexibilidad de Instalación y tareas de Administración, característico de la empresa Oracle.
- ✓ WebSphere presenta posibilidad de agrupamiento en clústeres, pero carece de soporte para el despliegue de aplicaciones en diferentes arquitecturas de procesador y plataformas de sistema operativo.
- ✓ JBoss ofrece una implementación ligera de JavaEE, muy aceptable para el desarrollo de aplicaciones de nivel mediano, esto resulta en la limitación de varias características para el despliegue de SOA.
- ✓ El presente estudio ha hecho posible elegir la mejor opción para la implementación del sistema de control de ventas de pasajes y encomiendas de TRANSPORTES PATRIA, el mismo que se ha elaborado utilizando el paradigma de SOA para facilitar su futura ampliación funcional y la integración con otras aplicaciones futuras.

RECOMENDACIONES

- ✓ SOA debe implementarse en las empresas como punto de partida para la elaboración de aplicaciones, ya que de esto depende su futuro mantenimiento y ampliación funcional.
- ✓ SOA debe tomar como un marco de referencia que permite a las organizaciones acoplarse a los nuevos requerimientos del mercado y de los clientes.
- ✓ El uso adecuado de un servidor de aplicaciones que permita la implementación de SOA ayuda a que las aplicaciones elaboradas tengan los tiempos de respuesta requeridos por la empresa.
- ✓ Un servidor de aplicaciones se debe tomar en cuenta que SOA debe permitirle Interoperabilidad entre los módulos de información de las Organizaciones, mediante APIs que permitan dicha conexión.
- ✓ Un servicio Web se debe implementar con tecnología JAX – WS la misma que describe qué debe contener un servicio web en Java.
- ✓ Un servicio WEB se debe implementar utilizando EJBs los mismos que permiten desarrollar aplicaciones empresariales de bajo acoplamiento.
- ✓ SOA se debe implementar utilizando el paradigma de desarrollo en N CAPAS, procurando la modularidad del sistema.
- ✓ Como futuro trabajo de Investigación se recomienda la "Orquestación de Aplicaciones SOA" para lograr una verdadera infraestructura de información en la empresa utilizando tecnologías como son ESB y BPEL.

RESUMEN

El Estudio Comparativo de Servidores de Aplicaciones (SA) para Desarrollo de Software con Arquitecturas Orientada a Servicios (SOA), sobre Plataformas JavaEE, se realizó con el propósito de evaluar características de interoperabilidad y tiempos de respuesta aplicados al sistema de Ventas de pasajes de Transportes Patria.

Se aplicó el método deductivo, analítico y se desarrolló el sistema mediante la metodología Microsoft Solutions Framework(MSF), se realizaron pruebas en máquinas virtuales para el despliegue de los servidores utilizando un sistema de BenchMarking. Para el desarrollo de la aplicación se utilizó Netbeans 6.9 y para generar reportes se empleó Ireport.

Se realizó el análisis de características entre los Servidores de Aplicaciones como Glassfish que obtuvo 88.33% frente a los 83.33% de Weblogic; en cambio WebSphere y jBoss obtuvieron 81.67%, luego se analizó la interoperabilidad en plataformas JavaEE corroborando a Glassfhis con 86.66% junto a WebLogic que obtuvieron el mayor porcentaje frente a 80% de WebSphere y 73.33% de Jboos, luego se simuló 3000 peticiones concurrentes de clientes a dichos servidores mediante Herramientas de BenchMarking, donde Glassfish obtuvo el menor tiempo de respuesta.

Se determina que el Servidor de Aplicaciones (SA) más idóneo en base a los indicadores planteados es Glassfish por sus características, interoperabilidad con plataformas JavaEE y su menor tiempo de respuesta a peticiones de clientes, en una Arquitectura Orientada a Servicios (SOA).

Las organizaciones y empresas deben elegir a Glassfish como SA ya que presenta altos niveles de agilidad, flexibilidad, escalabilidad e interoperabilidad, para adaptarse a los cambios de requerimientos de sus negocios, al dinamismo del mercado y a las demandas de sus clientes.

SUMMARY

The Comparative Applications Server Study (SA) for the Software Development with Architecture Oriented Services (SOA), on JavaEE Platforms was carried out to evaluate the inter-operation features and response times applied to the Transported Patria bus ticket sale.

The deductive and analytical method was applied and the system, though the Microsoft Solutions Framework methodology (MSF) was developed, Test in virtual machinery for the server display using the BenchMarking system were carried out. For the application development Netbeans6.9 was used and generating reports Ireport was used.

The analysis of the characteristics between Application Server such as Glassfish obtained 88.33% against 83.33% of WebLogic was carried out; on the other hendWebShephere and JBoss obtained 81.67%; then the inter-operation in the JavaEE platforms was analyzed corroborating Glassfish with 86.66% together withWebLogic which obtained the highest percentage against 80% of WebShephere and 73.33% of JBoss. The 3000 concurrent client petitions to such servers were simulated through BenchMarking Tools, where GlassFishhad the least response time.

It is determined that the most suitable Application Server (SA) based on the stated indicators is Glassfish because of its characteristics, interoperability with platforms JavaEE and its least response time to client petitions in an Architecture Oriented to Services (SOA).

The organizations and enterprise must choose Glassfish as SA as it present high levels of swiftness, flexibility, escalation and interoperability to adapt itself to change of business requirements, market dynamism and client demand.

GLOSARIO

SOA La Arquitectura Orientada a Servicios de cliente (en inglés **S**ervice **O**riented **A**rchitecture), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

JAVAEE Java Platform, Enterprise Edition o Java EE, es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

JVM Una Máquina virtual Java (en inglés Java Virtual Machine, JVM) es un máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java

XML XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

ESB En informática un bus de servicios de empresa (BSE) consiste en un combinado de arquitectura de software que proporciona servicios fundamentales para arquitecturas complejas a través de un sistema de mensajes (el bus) basado en las normas y que responde a eventos.

EJB Los Enterprise JavaBeans (también conocidos por sus siglas EJB) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales JAVAEE de Oracle Corporation (inicialmente desarrollado por Sun Microsystems). Su especificación detalla cómo los

servidores de aplicaciones proveen objetos desde el lado del servidor que son, precisamente, los EJB

JDBC Java Database Connectivity, más conocida por sus siglas JDBC, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice

SSL Secure Sockets Layer (SSL; protocolo de capa de conexión segura) y su sucesor Transport Layer Security (TLS; seguridad de la capa de transporte) son protocolos criptográficos que proporcionan comunicaciones seguras por una red, comúnmente Internet.

JSP JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

MYSQL MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones

LINUX Linux es, a simple vista, un Sistema Operativo. Es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo

BIBLIOGRAFIA

Arquitectura Orientada a Servicios

1. Arquitectura Orientada a Servicios (SOA) [en línea]

   http://es.scribd.com/doc/36397140/070717 Real World SOA

   2010/04/05

2. Arquitectura orientada a servicios [en línea].

   URL: http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

   2010/08/20

3. Arquitectura orientada a servicios [en línea].

   URL: http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

   2010/09/01

4. ORACLE SOA [en línea]

   http://www.oracle.com/us/technologies/soa/index.html

   2010/04/15

5. SOA [en línea]

   http://anheva 5.blogspot.com/2010/04/070717 real world soa view more.html

   2010/04/15

6. Servicios SOA [en línea]

   http://tecnologia.infobaeprofesional.com/notas/46399QueesSOAlaarquitecturaorientadaa servicios.html?

   2010/05/20

GlassFish

7. GlassFish[En línea]

   http://glassfish.java.net/

   2011/01/06

8. GlassFish[En línea]

http://es.wikipedia.org/wiki/GlassFish

2011/01/06

9. Java EE Community at a Glance GlassFish[En línea]

http://www.oracle.com/technetwork/java/javaee/community/index.html

2011/01/27

Servidores de Aplicaciones

10. Aplicaciones empresariales [en línea]

URL: http://h20223.www2.hp.com/services/cache/9269 0 0 140 470.html

2010/10/20

11. ¿Qué Es Un Servidor De Aplicaciones? [En línea]

URL:http://www.editum.org/QueEsUnServidorDeAplicacionesp473.html

20/08/2010

12. Servidores de Aplicaciones [en línea]

URL: http://www.alegsa.com.ar/Dic/servidor%20de%20aplicaciones.php

2010/05/20

13. Servidores de Aplicación [en línea]

URL: http://www.integraas.com/Servidores de Aplicacion.html

2011/01/12

14. ¿Qué es SOA, la arquitectura orientada a servicios? [En línea]

http://tecnologia.infobaeprofesional.com/notas/46399QueesSOAlaarquitecturaorientadaaservicios.html.

2011/01/12

JBOSS

15. Guía de JBoss [En línea]

http://www.osmosislatina.com/jboss/

2010/10/06

16. JBoss [En linea]

http://www.jboss.org/jbossas/downloads/

http://es.wikipedia.org/wiki/JBoss

2010/10/06

17. Primero pasos en JBoss[En línea]

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=JBossSeam_primeros_pasos

2010/10/06

Servicios Web

18. Introducción a los Servicios Web en Java [En línea]

http://www.programacion.com/articulo/introduccion_a_los_servicios_web_en_java_190

2011/01/24

19. Servicios Web en Java [En línea]

ttp://emilio.aesinformatica.com/2007/06/21/servicioswebenjavai/

2011/01/31

20. Servicios Web con JAXWS sin contenedores Java EE

http://willyxoft.wordpress.com/articulos/servicios_web_java_se/

2011/02/24

21. WS[En línea]

http://es.wikipedia.org/wiki/Servicio_web

2011/02/24

WebLogic

22. BEA WebLogic[En línea]

http://es.wikipedia.org/wiki/BEA_WebLogic

2010/10/06

23. BEA WebLogic

http://www.programacion.com/articulo/bea_weblogic:_introduccion_129

2010/10/06

24. Oracle WebLogic[En línea]

http://es.wikipedia.org/wiki/Oracle_WebLogic

2010/06/06

Websphere

25. IBM WebSphere [En línea]

http://en.wikipedia.org/wiki/IBM_WebSphere

20110301

http://www.gbm.net/software/websphere.php

2011/01/01

26. Introducción a WebSphere [En línea]

http://www.ibm.com/developerworks/ssa/websphere/newto/index.html

2010/06/06

27. Websphere[En línea]

http://www 01.ibm.com/software/websphere/#

http://es.wikipedia.org/wiki/WebSphere_%28software%29

2011/01/06

http://www.javahispano.org/forum/j2ee/es/websphere_3_5_7/

2011/01/06

ANEXOS

ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

FACULTAD DE INFORMATICA Y ELECTRONICA

ESCUELA DE INGENIERIA EN SISTEMAS

"MANUAL TÉCNICO DE ARTEMIA"

REALIZADO POR

WILSON JAVIER ROMERO GUILLEN

ANIBAL GIOVANY HERRERA MONCAYO

RIOBAMBA ECUADOR

2011

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Fase 1: Definición

1. Problema del desarrollo de la aplicación.

1.1. Definición del problema

La empresa de Transportes PATRIA necesita automatizar de manera eficiente el manejo de la información referente a sus actividades diarias, de tal manera que sus ejecutivos tengan la posibilidad de acceder a los datos relevantes en cualquier momento y por diferentes vías de comunicación.

Como punto inicial y motivo del presente trabajo de investigación se propone implementar el módulo de venta de Pasajes y Encomiendas, por tratarse del objetivo primordial de la empresa, pero con la posibilidad de ir agregando funcionalidades en el futuro y llegar a una verdadera plataforma de gestión de la información, sin entrar en procesos muy largos y costosos.

Además se necesita en la posteridad que la solución que se proponga permita la integración de las nueve sucursales de Transportes PATRIA en el país.

1.2. Visión del Proyecto

La empresa necesita de una aplicación elaborada con el paradigma de la Arquitectura Orientada a Servicios (SOA), para lograr que la información que se maneje en un aplicativo final pueda ser reutilizada al máximo en otros aplicativos futuros, sin entrar en complejos y largos procesos de ingeniería para implementarlos.

Los servicios SOA cumplen con estándares que facilitan la comunicación a través de protocolos de aceptación a nivel mundial, como ejemplo los Servicios WEB que utilizan el protocolo HTTP, que asegura que la información pueda ser accedida desde diferentes plataformas heterogéneas.

1.3. Perfiles de Usuarios

A continuación se define brevemente una descripción de cada uno de ellos:

Tabla I.1 Perfiles de Usuario

No.	Nombre	Perfil	Tipo de Acceso	Descripción
1	Administrador	Administrador del sistema.	Escritorio	Permiso para realizar el registro, eliminación y actualización de cuentas de usuario, dependencias, Frecuencias, Socios para tareas especificas del sistema
2	Usuario	Usuario del sistema	Escritorio	Permiso únicamente para registrar la venta de pasajes o encomiendas y emitir el respectivo comprobante.
3	Gerente	Usuario del sistema	Web	Permiso para revisar la información que suministra el sistema a través de reportes e informes.
4	Cliente	Usuario del Sistema	Web	Permiso para revisar los horarios de frecuencias y consultar datos de envíos por encomiendas.

1.4. Ámbito del Proyecto

1.4.1. Alcance

Se propone la elaboración de una solución integrando diferentes ambientes y tecnologías para solventar las necesidades de los solicitantes. Se elaborará una base de datos con soporte de las operaciones solicitadas, además una aplicación de escritorio que permitirá el registro de la venta de pasajes y encomiendas.

Una aplicación Web que permitirá varias consultas de Información relevante a nivel gerencial, y una aplicación de escritorio que permitirá la administración del sistema en cuanto a cuentas de usuario y los permisos de acceso.

1.4.2. Requerimientos Generales

Tabla I.2 Requerimientos Generales

Número	Requerimiento
REQ1	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a cuentas de usuario del sistema.
REQ2	El sistema deberá permitir altas, bajas y actualizaciones de la información que corresponde a las funciones permitidas para cada usuario.
REQ3	El sistema deberá permitir el registro de la venta de pasajes diarios.
REQ4	El sistema deberá permitir el registro de la venta de encomiendas.
REQ5	El sistema deberá permitir la consulta de información a través de un portal web, que permita la obtención de información relevante para el nivel gerencial.
REQ6	El sistema deberá permitir la consulta de las frecuencias disponibles a los clientes.
REQ7	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a dependencias.
REQ8	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a las frecuencias.
REQ9	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a socios.
REQ10	El sistema deberá permitir la consulta de información disponible sobre encomiendas de los clientes.

1.5. Concepto de la Solución

Se plantea la elaboración de una solución con arquitectura Orientada a Servicios ya que este paradigma facilitará el objetivo que tiene la empresa de ir ampliando su plataforma incorporando nuevas funcionalidades y módulos independientes, siempre cuidando que la información pueda ser accedida y compartida sin entrar en procesos complejos de análisis e ingeniería.

Se elaborará una base de datos relacional para respaldar y registrar la información que se genera durante el desarrollo de las actividades de la empresa.

SOA estará presente en la aplicación obligatoriamente implementado a través de servicios WEB para aprovechar la ventaja del transporte de la información mediante el protocolo HTTP, ampliamente aceptado en distintos dispositivos y arquitecturas.

Como plataforma de desarrollo se utilizará JavaEE para asegurar la portabilidad de las aplicaciones entre distintos Sistemas Operativos, del ambiente móvil y de escritorio.

Se elaborará una aplicación de Escritorio en el ambiente Java Swing destinada a la venta de Pasajes y Encomiendas que obtendrá la información para sus tareas de los servicios WEB disponibles.

Las consultas de información Relevante para el Nivel Gerencial de la empresa se harán mediante una aplicación WEB.

Las tareas de administración del sistema se realizarán mediante una aplicación de escritorio, misma que tendrá el acceso permitido solo al personal autorizado.

1.5.1. Uso de Herramientas

Tabla I.3 Herramientas Utilizadas

Característica Técnica	Detalle
Plataforma Operativa de Funcionamiento	La solución planteada es independiente de la plataforma, por lo que se puede usar cualquier sistema operativo con soporte JAVA.
Motor de Base de Datos	Se utilizará el motor de base de datos MYSQL 5 el cual brinda la posibilidad de elaborar Bases de Datos Relacionales y el manejo de procedimientos almacenados para las diferentes transacciones.
Herramientas de Desarrollo	Entre las Herramienta de Desarrollo se ha seleccionado el IDE NETBEANS 6.9 que permite el trabajo sobre la plataforma JAVA y la posibilidad de implementar servicios web de forma fácil, lo que aumenta la productividad de nuestro equipo de desarrollo.
Manejador de Reportes	Los reportes se manejarán utilizando JASPER y como entorno para el diseño de los mismos la herramienta I-REPORT. Además se elaborarán reportes dinámicos con la suite PENTAHO.
Herramientas de Modelado UML	Para la elaboración de ciertos tipos de diagramas que permitirán el análisis se utilizará STAR UML versión 5.
Herramientas de oficina	Se utilizará la suite OPEN OFFICE para la elaboración de la documentación básica sobre el desarrollo del proyecto.
Herramientas para Administración **y** diseño de base de Datos	Se utilizará la suite de MYSQL, entre estas MySql GUI Tools 5.02 y MySql Work Bench 5.02. Además se utilizará SQLyog como apoyo al desarrollo de la base de datos.
Servidor de Aplicaciones	Se utilizará GLASSFISH v3.1, que nos ha resultado la mejor opción a partir de la presente investigación.
Sistema Operativo Servidor	Se utilizará como Servidor a Linux CentOS 5.4 para la Base de Datos y el Servidor de Aplicaciones elegido.

1.5.2. Planteamiento de la Arquitectura de la Aplicación

Tabla I.4 Arquitectura de la Aplicación

INTERFAZ DE USUARIO (ESCRITORIO / WEB / MOVIL) (CAPA 5)
SERVICIOS WEB (CAPA 4)
REGLAS DE NEGOCIO (CAPA 3)
ACCESO A DATOS (CAPA 2)
BASE DE DATOS (CAPA 1)

Se propone una arquitectura que consta de 5 capas:

Tabla I.5 Capas de la Aplicación

CAPA 1: BASE DE DATOS	En el que se elaborará una base de datos relacional, para soportar la lógica del negocio. Aquí se brindará acceso a la información a través de procedimientos almacenados, debido a su rapidez de ejecución.
CAPA 2: ACCESO A DATOS	El acceso a datos representará a todas las clases propuestas para el sistema y se implementará utilizando JDBC.
CAPA 3: REGLAS DE NEGOCIO	Las reglas de negocio utilizarán los servicios de la capa de acceso para efectuar los procesos requeridos y presentar la información al usuario final.
CAPA 4: SERVICIOS WEB	Mediante la implementación de servicios web se brinda la posibilidad de brindar escalabilidad a la aplicación y la facilidad de acceso a la información segura a través de internet utilizando protocolos de comunicación segura. Estas características son propias de SOA
CAPA5: INTERFACES DE ESCRITORIO/ WEB/MOVIL	Esta capa utilizará los servicios web publicados y desplegará la información al usuario. También se efectuará la alimentación de la información a través de esta hacia la base de datos.

1.6. Objetivos del Proyecto

1.6.1. Objetivos del Negocio

- Administrar la información centralizadamente, identificando fácilmente los registros de información.
- Implementar la base de una estructura funcional que permita escalabilidad sin entrar en procesos complejos y costosos de ingeniería.
- Manejar la información de manera eficiente y en el menor tiempo posible.

1.6.2. Objetivos del Diseño

- Lograr que la información que se maneje dentro de la organización pueda ser consultada solo por el personal autorizado.
- Disminuir el tiempo de operación y respuesta en la ejecución de los procesos.
- Brindar interfaces de usuario amigables e intuitivas que faciliten el trabajo cotidiano de los encargados en los diferentes procesos del negocio.

1.7. Factores Críticos – Análisis de Riesgos

Análisis de Riesgos

Identificación del Riesgo

RN: Riegos de Negocios.
RT: Riesgos Técnicos.
RP: Riesgos de Proyectos

Tabla I.6 Tabla de Riesgos

Id	Descripción del Riesgo	Categoría	Consecuencia
R1	Falta de especificación de requerimientos necesarios.	**RT**	▪ Retraso en el proyecto. ▪ Pérdida de Tiempo.
R2	Interfaz de usuario mal definida.	**RT**	▪ El proyecto no presta la funcionalidad deseada.
R3	El tiempo de Desarrollo es muy corto.	**RN**	▪ El producto final no es totalmente funcional.
R4	Falta de personal necesario para el desarrollo del proyecto.	**RP**	▪ La solución no se completa ▪ Solución planteada erróneamente
R5	Inestabilidad del personal técnico en el desarrollo de sus funciones	**RT**	▪ Los componentes desarrollados no interactúan correctamente. ▪ Cambio de responsabilidades sobre la marcha del proyecto. ▪ Pérdida de tiempo.
R6	Reutilización del software en menor grado de la prevista.	**RP**	▪ Empleo de mayor tiempo en el desarrollo de nuevos componentes.
R7	El diseño de la Base de datos no cumple con los requerimientos funcionales	**RT**	▪ Cambios en la base de datos sobre la marcha del proyecto ▪ Pérdida de Tiempo.
R8	Las herramientas de desarrollo seleccionadas no brindan la funcionalidad prevista.	**RT**	▪ Empleo de mayor tiempo en el cumplimiento de las

			funciones.
R9	Planificación y distribución de Responsabilidades erróneas.	**RP**	▪ Entrega de un proyecto parcialmente funcional. ▪ Reasignación de Funciones ▪ Pérdida de Tiempo
R10	La distribución del tiempo asignado a las tareas no se cumple.	**RP**	▪ Retraso en el desarrollo del proyecto.

Análisis del Riesgo

Valoración de la Probabilidad.

Tabla I.7 Valoración de la Probabilidad

Porcentaje	Descripción	Valor
1% - 33%	Baja	1
34% - 67%	Media	2
68% - 99%	Alta	3

Valoración del Impacto

Tabla I.8 Valoración del Impacto

Impacto	Costo	Retraso	Impacto Técnico.	Valor
Bajo	< 1%	1 semana	Ligero efecto en el desarrollo del proyecto.	1
Moderados	< 5%	2 semanas	Moderado efecto en el desarrollo del proyecto.	2
Alto	< 10%	1 mes	Severo efecto en el desarrollo del proyecto	3
Crítico	> 10%	> 1 mes	El proyecto no puede ser culminado.	4

Valoración de la Exposición del Riesgo.

Tabla I.9 Valoración de la Exposición del Riesgo

Exposición del Riesgo	Valor	Color
Baja	1 o 2	Verde
Media	3 o 4	Amarillo
Alta	>= 6	Rojo

Impacto / Probabilidad	Bajo=1	Moderado=2	Alto=3	Crítico=4
Alta=3	3	6	9	12
Media=2	2	4	6	8
Baja=1	1	2	3	4

Identificación	Probabilidad			Impacto		Exposición al Riesgo	
	%	Valor	Probabilidad	Valor	Impacto	Valor	Expo
R7	75	3	Alta	3	Alta	9	Alta
R4	40	2	Media	4	Crítico	8	Alta
R1	40	2	Media	3	Alta	6	Alta
R3	50	2	Media	3	Alta	6	Alta
R9	15	1	Baja	3	Alta	4	Media
R5	15	1	Baja	3	Alta	4	Media
R6	15	1	Baja	3	Alta	4	Media
R2	40	2	Media	2	Moderado	4	Media
R10	50	2	Media	2	Moderado	4	Media
R8	20	1	Baja	1	Bajo	1	Baja

Línea De Corte

1.8. Plan de reducción, supervisión y gestión del riesgo.

Consideramos los riesgos de alto impacto.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R10		**FECHA: 2010-10-15**	
Probabilidad: Media **Valor:2**	**Impacto:** Moderado **Valor:** 2	**Exposición:** Media **Valor:** 4	**Prioridad:** Media **Valor: 2**
DESCRIPCIÓN: La distribución del tiempo asignado a las tareas no se cumple.			
REFINAMIENTO: Causa1: Falla en el proceso de interconexión de equipos y comunicación de servidores para el desarrollo. Causa2: Cambios en el diseño planteado inicialmente. Consecuencia: Retraso en el cumplimiento de las tareas previstas para el tiempo determinado.			
REDUCCIÓN Y SUPERVISIÓN: <u>Reducción:</u> ✓ ……..Preparar los equipos y someterlos a distintas pruebas antes del comienzo del proyecto. ✓ ……..Implementar un diseño con una visión amplia del producto terminado. <u>Supervisión:</u> ✓ ……..Revisar que los planes de contingencia contra las causas previstas sean cumplidos.			
GESTIÓN: Elaborar planes de contingencia para evitar estos inconvenientes durante la marcha del proyecto			
ESTADO ACTUAL: Fase de reducción iniciada [ ] Fase de Supervisión iniciada [ ] Gestionando el Riesgo. [X]			
RESPONSABLES: Aníbal Herrera, Javier Romero			

1.9. Planificación Inicial

RECURSOS HUMANOS

INTEGRANTES Y SUS FUNCIONES.

Tabla I.10 Integrantes y sus Funciones

Nombre	Función
Aníbal Herrera	Administrador del Producto
Javier Romero	Administrador del Programa
Javier Romero, Anibal Herrera	Equipo de desarrollo
Aníbal Herrera, Javier Romero	Equipo analista de la Experiencia del Usuario
Javier Romero	Equipo analista de Pruebas y Calidad
Aníbal Herrera	Equipo encargado del despliegue del Proyecto

RECURSOS FISICOS (SW / HW)

Tabla I.11 Recursos Físicos (HW / SW)

Numero	Detalle
2	Equipos Intel Core Duo 2.4 GHZ
1	Sistema operativo con soporte JAVA.
1	Entorno de desarrollo Integrado NETBEANS 6.9
1	Servidor de Aplicaciones GLASSFISH v3.1
1	Servidor de Base de Datos MYSQL v5
1	Suite de Oficina Open Office.
1	Suite de Inteligencia de Negocios PENTAHO

1.10. PLANIFICACION DE ACTIVIDADES

Tabla I.12 Planificación de Actividades

ACTIVIDADES	Sept.	Oct.	Nov.	Dic.	Ene.	Feb.	RESPONSABLES
	1	2	3	4	5	6	
DEFINICIÓN DEL PROBLEMA	X						Aníbal Herrera
CONCEPTO DE LA SOLUCIÓN	x						Aníbal Herrera, Javier Romero.
DEFINIR LOS REQUERIMIENTOS DEL SISTEMA.		X					Aníbal Herrera, Javier Romero.
DEFINICION DE CASOS DE USO.		X					Aníbal Herrera
DEFINICION DEL MODELO LOGICO (CLASES Y DATOS).		X					Javier Romero.
DEFINIR EL MODEL FISICO (COMPONENTES Y DATOS).			X				Aníbal Herrera
DESARROLLO (INTERFAZ, ACCESO A DATOS NEGOCIO).			X	x	X		Javier Romero.
PRUEBAS						X	Aníbal Herrera.
DESPLIEGUE						X	Aníbal Herrera, Javier Romero.

Fase 2: Planificación

2. Planificación

2.1. Especificación Funcional

2.1.1. Diseño Conceptual

2.1.1.1. Requerimientos

REQUERIMIENTOS FUNCIONALES

En este apartado se exhiben los requisitos que el sistema deberá satisfacer

Tabla II.1 Requerimientos Funcionales

Número	Requerimiento
REQ1	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a cuentas de usuario del sistema.
REQ2	El sistema deberá permitir altas, bajas y actualizaciones de la información que corresponde a las funciones permitidas para cada usuario.
REQ3	El sistema deberá permitir el registro de la venta de pasajes diarios.
REQ4	El sistema deberá permitir el registro de la venta de encomiendas.
REQ5	El sistema deberá permitir la consulta de información a través de un portal web, que permita la obtención de información relevante para el nivel gerencial.
REQ6	El sistema deberá permitir la consulta de las frecuencias disponibles a los clientes.
REQ7	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a dependencias.
REQ8	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a las frecuencias.
REQ9	El sistema deberá permitir altas, bajas y actualizaciones de la información correspondiente a socios.
REQ10	El sistema deberá permitir la consulta de información disponible sobre encomiendas de los clientes.
REQ 11	El sistema deberá presentar opciones de seguridad para el ingreso mediante cuentas de usuario y perfiles de acceso.

REQ 12	El sistema deberá permitir la obtención de la lista de pasajeros de una unidad y frecuencia determinada.

REQUERIMIENTOS NO FUNCIONALES

Aquí se muestran los requerimientos no funcionales, que a su vez son las características principales del sistema a desarrollar:

Tabla II.2 Requerimientos no Funcionales

Requerimiento	Detalle
Compatibilidad	Se necesita que el sistema brinde la posibilidad de ser accedido desde navegadores web de distintas plataforma y fabricantes.
Fácil de Manejar	La interfaz de usuario debe ser intuitiva para que de esta forma los usuarios nuevos del sistema se adapten rápidamente.
Fiabilidad	El sistema deberá manejar soluciones de respaldo en caso de desperfectos, permitiéndole estar siempre disponible.
Estabilidad	El sistema deberá estar configurado de tal forma que no presente situaciones de desperfecto continuas que afecten al desarrollo del trabajo continuo.
Seguridad	El acceso a la información pertinente de cada perfil de usuario se debe manejar utilizando sesiones y sistemas de autenticación y autorización.
Integridad	La información debe ser alimentada a la base de datos pasando por procesos de validación.
Flexibilidad	La aplicación deberá permitir diferentes entornos de configuración presentando la facilidad de migración de plataforma y escalabilidad.
Fácil Instalación	Los componentes del sistema no presentan dificultades para su instalación ya que en su mayoría ofrecen la posibilidad de instalación y configuración desatendida.

2.1.1.2. Actores

Tabla II.3 Actores

No.	Nombre	Perfil	Tipo de Acceso	Descripción
1	Administrador	Administrador del sistema	Vía Escritorio	Realiza altas, bajas, actualizaciones de cuentas de usuario, socios y frecuencias.
2	Cajera	Usuario del Sistema	Vía Escritorio	Realiza altas, bajas y actualizaciones de remitentes y clientes. Realiza el registro de la venta de pasajes y encomiendas.
3	Gerente	Usuario del Sistema	Vía Web	Realiza consultas de datos relevantes sobre frecuencias realizadas, Unidades de transporte, Socios, Cantidades sobre ventas. Realiza el registro de la planificación de Frecuencias.
2	Cliente	Usuario del sistema	Vía Web	Realiza consultas de Frecuencias disponibles y datos de encomiendas.

2.1.1.3. Casos de Uso y Escenarios

Titulo:	**Administración del sistema**

Diagrama:

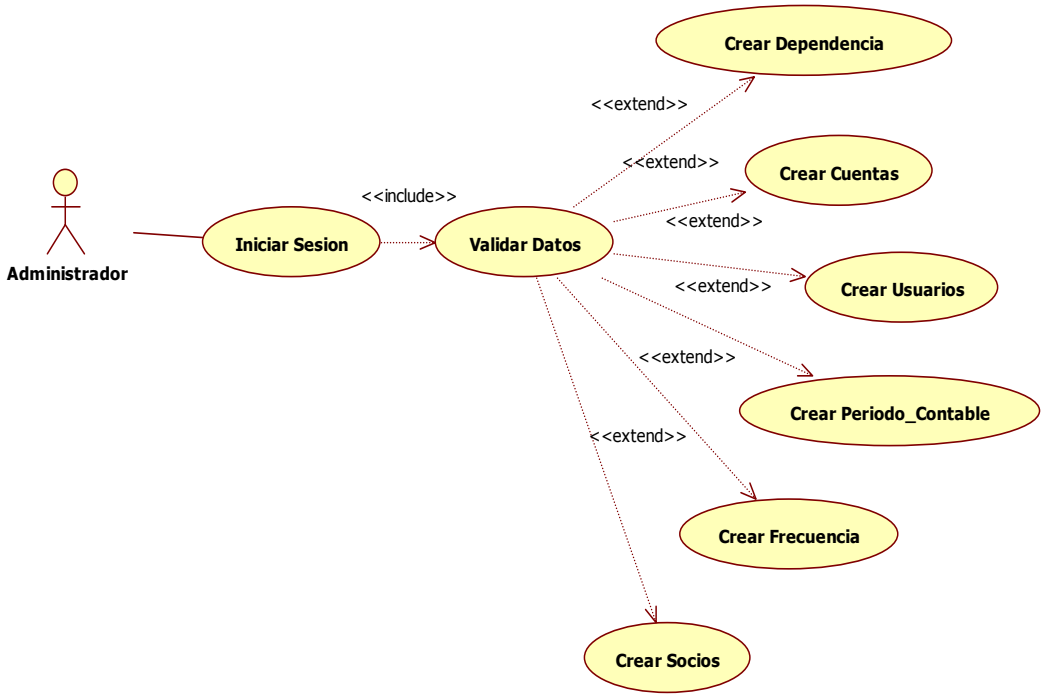


Figura II.1 Caso de Uso Administrador

Número de REQ: 2	**Usuario:** Administrador

Nombre historia: Administración del sistema	
Prioridad en negocio:1	**Riesgo en desarrollo:** 3
Puntos estimados:	**Iteración asignada:**
Programador responsable: Javier Romero, Anibal Herrera	

Descripción:

- …. El administrador del sistema inicia sesión con un nombre de usuario y contraseña.
- ….Posteriormente se le presentan las funciones a las que tiene permitido ingresar.
- ….El administrador realiza el registro de la información correspondiente a cuentas de usuario para el acceso al sistema, dependencias, unidades, frecuencias y socios.

Curso Típico de Eventos:

ACCIONES DE ACTORES	RESPUESTAS DEL SISTEMA
El administrador inicia la aplicación.	El sistema le presenta la pantalla de inicio de sesión.
El administrador ingresa las credenciales de acceso.	El sistema verifica las credenciales y en caso de ser válidas le permite el ingreso al sistema.
El administrador realiza el ingreso de los datos de las nuevas cuentas.	El sistema presenta el formulario correspondiente para luego enviar esta información a la base de datos.
El administrador realiza la actualización de los datos de las cuentas de usuario.	El sistema presenta el formulario correspondiente para luego enviar esta información a la base de datos.
El administrador realiza la eliminación de los datos de cuentas de usuario.	El sistema efectúa la transacción hacia la BD.

Observaciones:

Procedimiento similar para el registro de socios, dependencias y unidades.

Titulo:	Registro de la información por parte del cajero(a).

Diagrama:

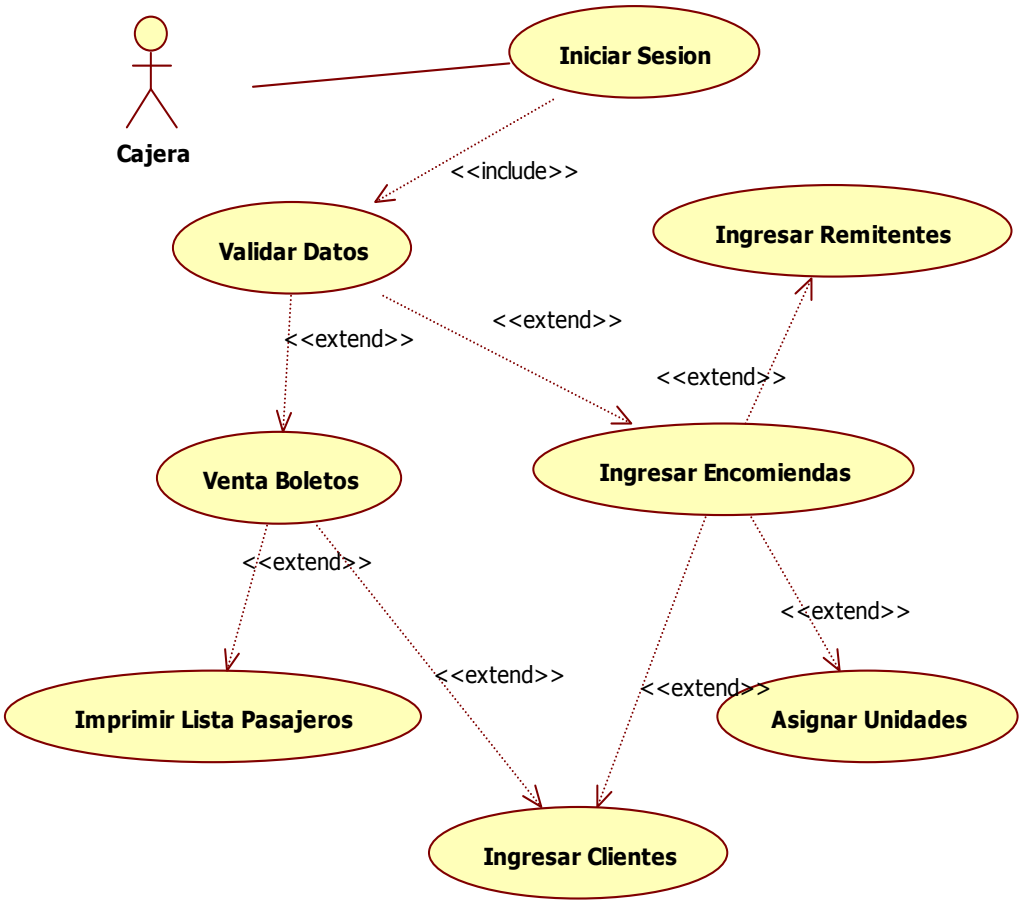


Figura II.2 Caso de Uso Cajero(a)

Número de REQ: 7	**Usuario:** Cajero(a)

Nombre historia: Registro de la información por parte del cajero(a).	
Prioridad en negocio: 1	**Riesgo en desarrollo:** 2
Puntos estimados:	**Iteración asignada:**
Programador responsable: Javier Romero, Anibal Herrera	

Descripción:

- …. El cajero(a) ingresa al sistema con su nombre de usuario y contraseña para realizar las tareas de registro de ventas de pasajes y encomiendas.
- ….El cajero(a) puede también realizar el registro de Remitentes y asignación de unidades para encomiendas, agregar clientes e imprimir la lista de pasajeros de una unidad y frecuencia determinada.

Curso Típico de Eventos:

ACCIONES DE ACTORES	RESPUESTAS DEL SISTEMA
El cajero(a) inicia la aplicación.	El sistema le presenta la pantalla de inicio de sesión.
El cajero(a) ingresa las credenciales de acceso.	El sistema verifica las credenciales y en caso de ser válidas le permite el ingreso al sistema.
El cajero(a) realiza el ingreso de los datos correspondientes a ventas de pasajes y encomiendas.	El sistema presenta el formulario correspondiente para luego enviar esta información a la base de datos.
El usuario realiza la actualización de los datos correspondientes a ventas diarias.	El sistema presenta el formulario correspondiente para luego enviar esta información a la base de datos.
El usuario realiza la eliminación de datos correspondientes a ventas diarias.	El sistema efectúa la transacción hacia la BD.

Observaciones:

El proceso de registro de remitentes y clientes se hace de forma similar.

Titulo:	Consulta de información por parte del gerente.

Diagrama:

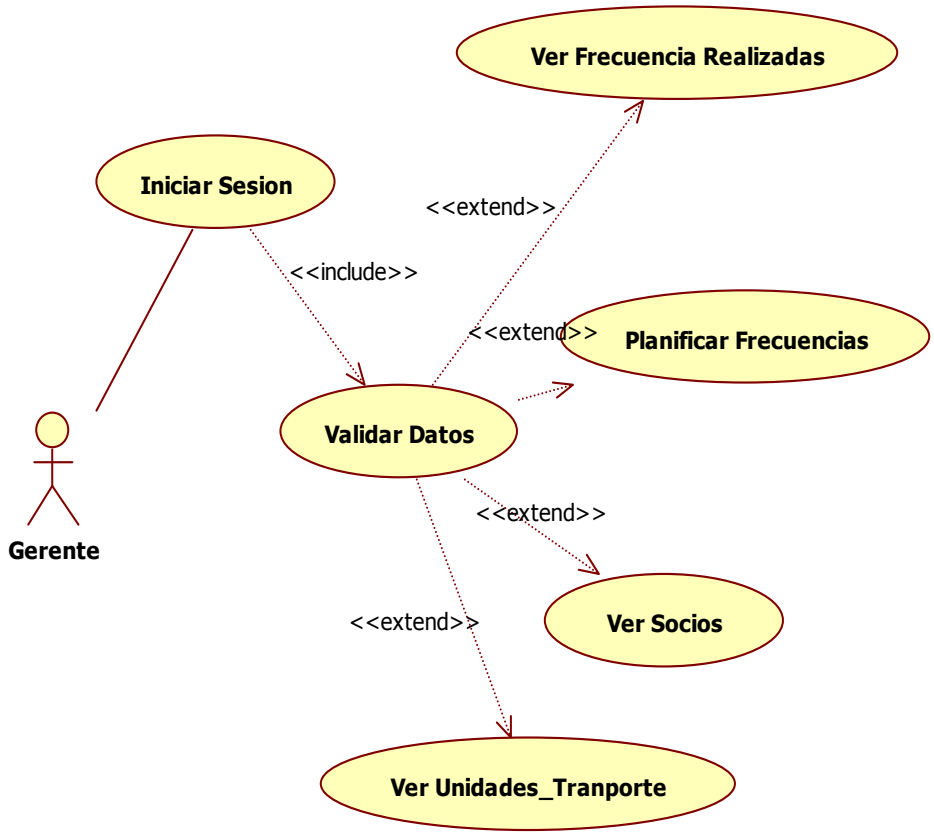


Figura II.3 Caso de Uso Gerente

Número de REQ: 6	**Usuario:** Gerente

Nombre historia: Consulta de información por parte del gerente	
Prioridad en negocio: 1	**Riesgo en desarrollo:**3
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Javier Romero, Anibal Herrera	

Descripción:

- …. El gerente ingresa al sistema proporcionando las credenciales de acceso.
- …. El sistema valida la información y presenta las opciones de consulta para el gerente. Todos estos datos son de carácter confidencial, por lo que solo se permite el acceso al personal autorizado. El gerente puede revisar información sobre frecuencias, socios, unidades de transporte, cantidades de ventas. Adicionalmente se puede efectuar el ingreso del calendario de frecuencias.

Curso Típico de Eventos:

ACCIONES DE ACTORES	RESPUESTAS DEL SISTEMA
El gerente ingresa a la aplicación web.	El sistema le presenta la pantalla de inicio de sesión.
El gerente ingresa las credenciales de acceso.	El sistema verifica las credenciales y en caso de ser válidas le permite el ingreso al sistema.
El gerente realiza la consulta de las frecuencias realizadas.	El sistema presenta la información solicitada.
El gerente realiza el registro del calendario de frecuencias.	El sistema presenta el formulario correspondiente para luego enviar esta información a la base de datos.
El gerente realiza la consulta de unidades de transporte.	El sistema presenta la información solicitada.

Observaciones:

Todas las consultas de información llevan un proceso similar.

Titulo:	**Consulta de información por parte del cliente.**

Diagrama:

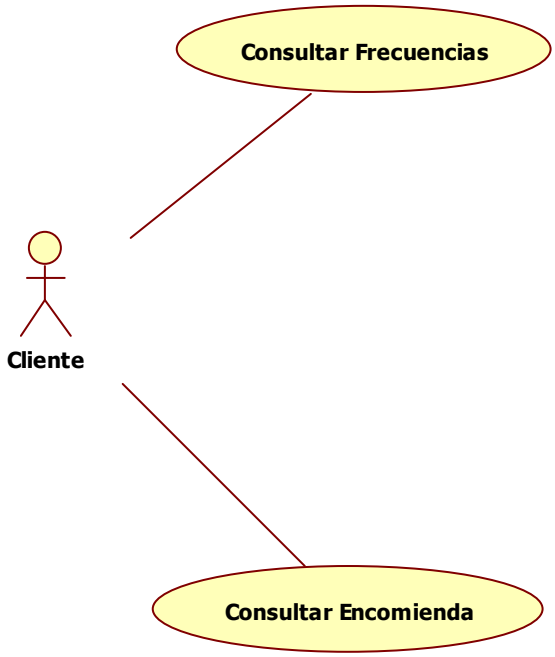


Figura II.4 Caso de Uso Cliente

Número de REQ: 6	**Usuario:** Gerente

Nombre historia: Consulta de información por parte del cliente	
Prioridad en negocio: 1	**Riesgo en desarrollo:**3
Puntos estimados:1	**Iteración asignada:**1
Programador responsable: Javier Romero, Anibal Herrera	

Descripción:

- …. El cliente ingresa al sistema de consultas vía web.
- …. El sistema valida la información y presenta las opciones de consulta para el cliente. El cliente puede revisar la información pública a cerca de las frecuencias planificadas y los envíos por encomiendas.

Curso Típico de Eventos:

ACCIONES DE ACTORES	RESPUESTAS DEL SISTEMA
El cliente ingresa a la aplicación web.	El sistema le presenta la pantalla de consultas.
El cliente ingresa los parámetros de consulta.	El sistema verifica los parámetros y presenta la información solicitada.
El cliente realiza la consulta de las frecuencias disponibles.	El sistema presenta la información solicitada.
El cliente realiza la consulta de las encomiendas realizadas.	El sistema presenta la información solicitada.

Observaciones:

Todas las consultas de información llevan un proceso similar.

2.1.1.4. Glosario de Términos

Tabla II.4 Glosario

TERMINO	DESCRIPCION
FAST SOLUTIONS	Empresa de desarrollo creada para ofrecer soluciones informáticas de calidad, innovadores en el campo del desarrollo de Aplicaciones Informáticas de apoyo logístico empresarial.
MSF	Microsoft Solution Framework, metodología ágil empleada para el desarrollo de la aplicación.
Servidor de Base de Datos	El servidor de la base de datos es un programa de computadora que proporciona servicios de la base de datos a otros programas o computadoras, puede también referirse a una computadora dedicada a funcionar tal programa.
Base de Datos	Cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora, diseñado para facilitar su mantenimiento y acceso de una forma estándar.
Conexión	Comunicación entre dos entes que tienen características similares.
Interfaz	Punto en el que se establece una conexión entre dos elementos, que les permite trabajar juntos. La interfaz es el medio que permite la interacción entre esos elementos.
Sistema Operativo	Software básico de una computadora. Tiene tres grandes funciones: coordina y manipula el hardware del ordenador o computadora, organiza los archivos en diversos dispositivos de almacenamiento y gestiona los errores de hardware y la pérdida de datos

JAVA	Plataforma de desarrollo que posibilita la creación de aplicaciones SOA de tipo empresarial, interoperables e independientes de la arquitectura de despliegue.
SOA	Arquitectura Orientada a Servicios.
SA	Servidor de Aplicaciones, software que interactúa entre la aplicación cliente y el motor de base de datos para el acceso a la información.
Aplicación WEB	Aplicación que posibilita su acceso mediante un navegador web y conexión a internet. Utiliza el protocolo HTTP.
Aplicación de Escritorio	Aplicación que permite el acceso a la información mediante formularios y cuadros de diálogo, también puede obtener su información de internet.
Aplicación Móvil	Formularios apropiados para dispositivos celulares que permiten el acceso a la información también mediante una conexión de internet en el equipo.
UML	Lenguaje de Modelado Universal, se utiliza para la descripción de los procesos y la ingeniería del sistema.

2.1.2. Diseño Lógico

2.1.2.1. Tecnología a utilizar en el Proyecto

Herramientas:

De acuerdo a la naturaleza de los requerimientos y la solución planteada para estos se ha elegido a la tecnología JAVA para la implementación del proyecto, debido a que las aplicaciones desarrolladas con esta tecnología son independientes de la plataforma y posibilitan el despliegue de Servicios Web para el empleo de SOA.

Como metodología de desarrollo se ha elegido MSF como referencia, por cuanto es adaptable a cualquier tipo de proyecto, sea este grande o pequeño, y también se adapta al tiempo disponible para el desarrollo.

Aplicación Multicapas:

Almacenes de Datos:

MYSQL nos permitirá la implementación de los objetos necesarios para el soporte de la solución. Principalmente nos permite la elaboración de procedimientos almacenados que brindan mayor rapidez en la ejecución y facilidad de mantenimiento.

Capa de acceso a Datos:

El acceso a datos se implementó utilizando JDBC de JAVA ya que nos permite la abstracción de la mayoría de transacciones que representan la ejecución de sentencias y la devolución de conjuntos de datos para el procesamiento interno de la aplicación.

Capa de lógica de Negocio:

La lógica de negocio se implementará utilizando clases JAVA elaborando componentes que permitan realizar las tareas funcionales de la empresa.

Capa de Servicios Web:

La capa de servicios web se implementó utilizando componentes JAVA BEANS, los cuales son métodos públicos que pueden ser consumidos por las aplicaciones cliente propuestas.

Capa de Presentación o Interfaz de Usuario:

La aplicación Web se implementará utilizando JSP de JAVA para asegurar que el estándar permita posteriormente migrar la aplicación a plataformas distintas a las que se propuso su implementación. La aplicación de escritorio de igual forma puede ser ejecutada sobre distintas plataformas de SO y se elaboró mediante la utilización de formularios Java SWING.

2.1.2.2. Diagramas de Secuencia

Título:	Administración del Sistema

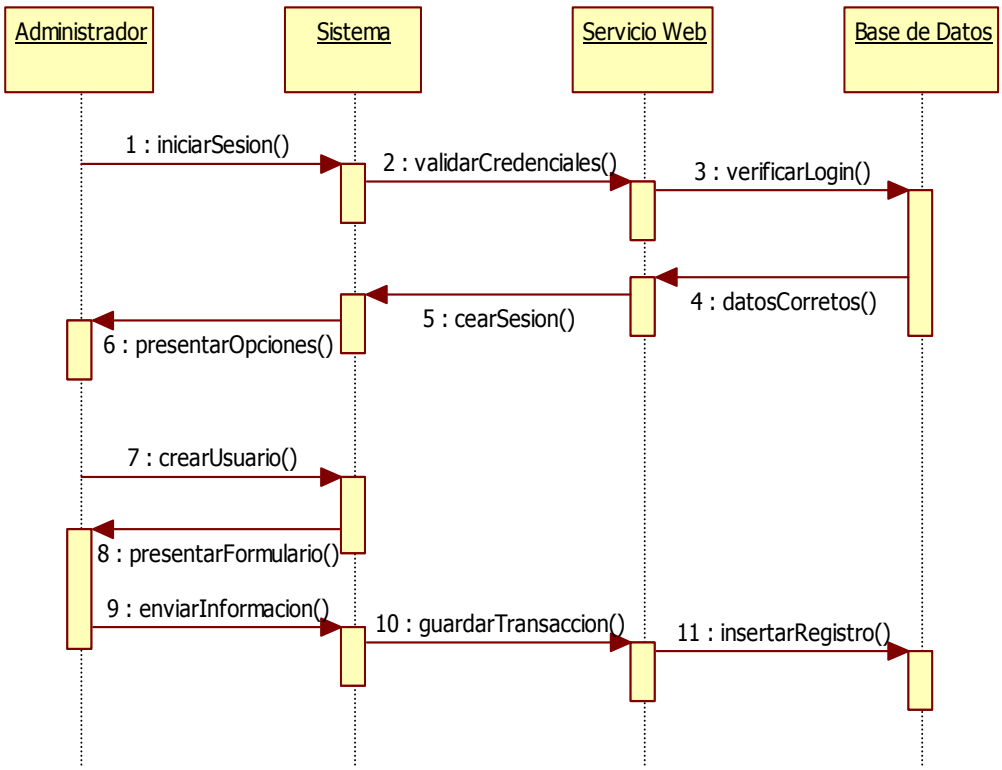


Figura II.5 Diag. Secuencia Administrador

Título:	Registro Venta de Pasaje

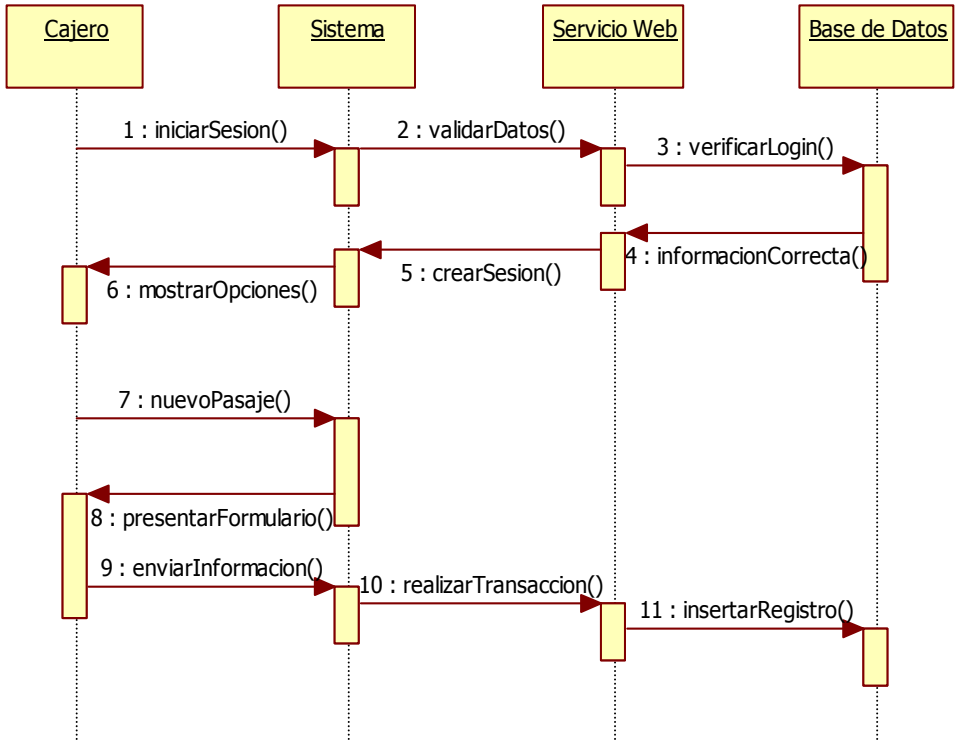


Figura II.6 Diag. Secuencia Cajero

Título:	Consultas Gerenciales

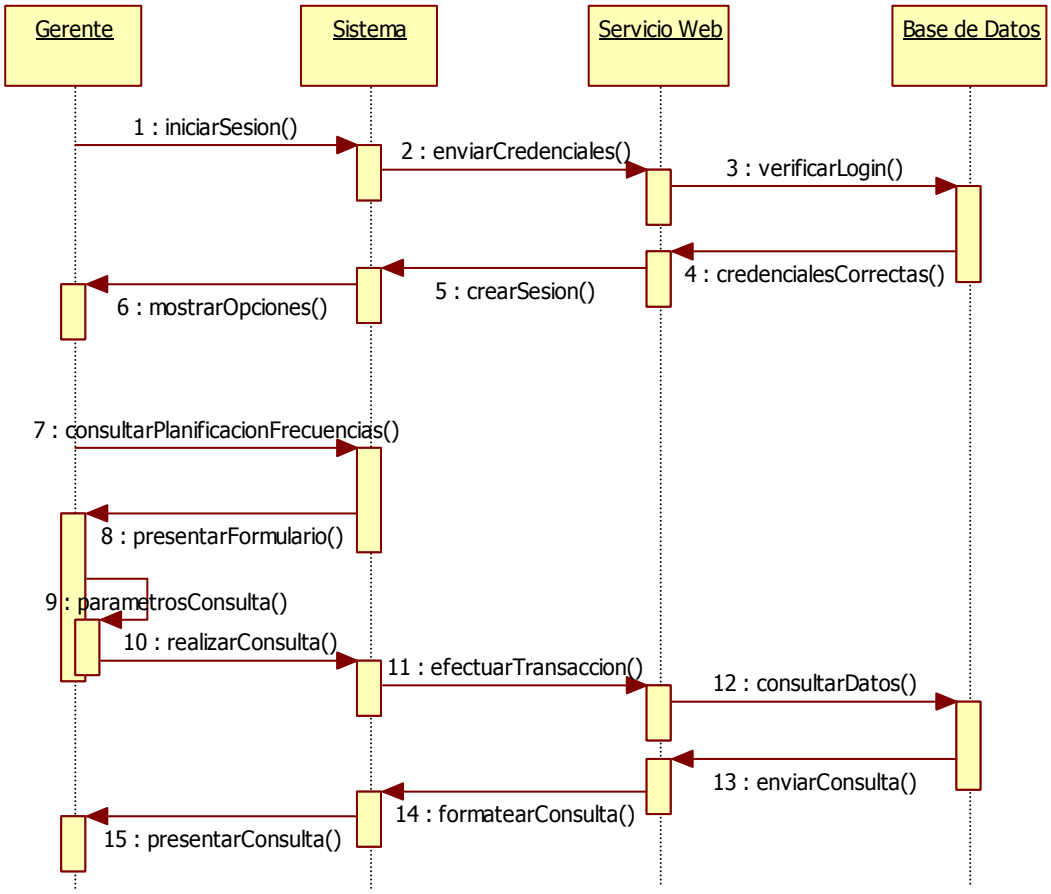


Figura II.7 Diag. Secuencia Gerente

Título:	Consultas Clientes

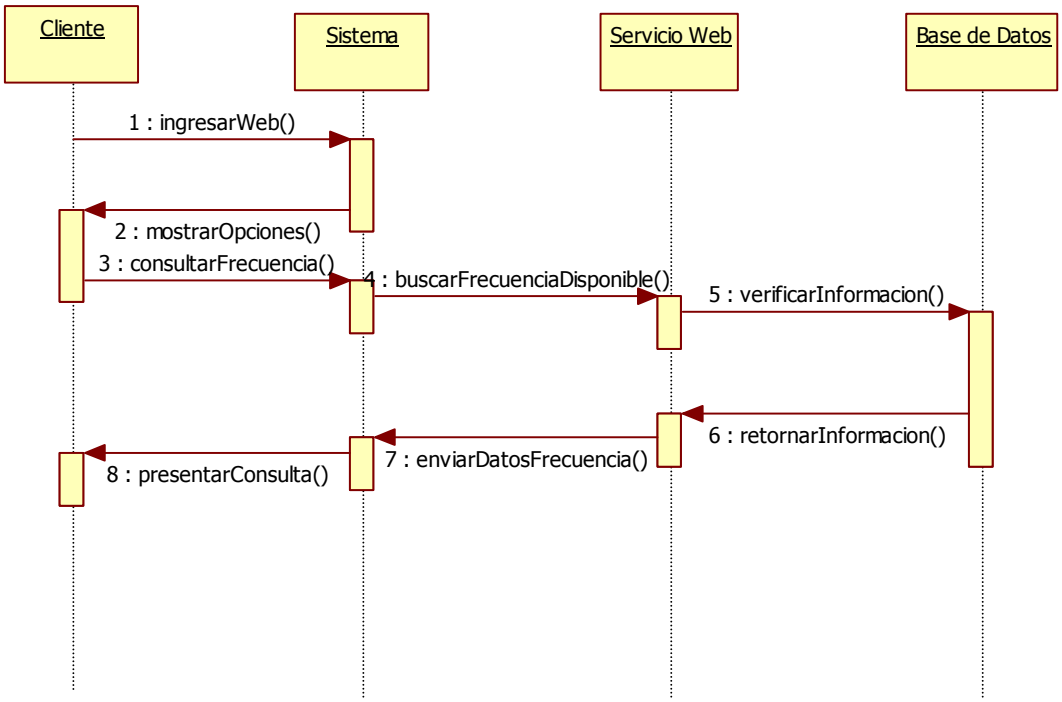
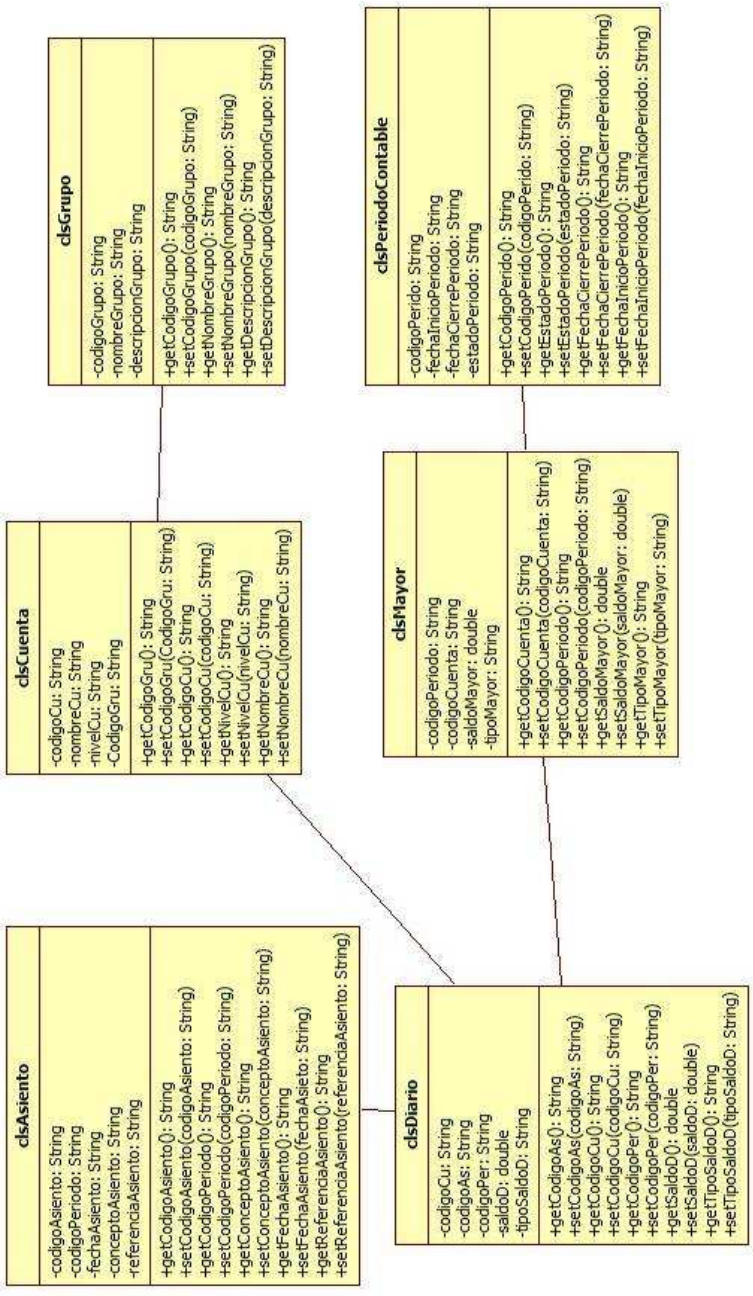


Figura II.8 Diag. Secuencia Cliente

2.1.2.3. Diagrama de Clases

Título:	Diagrama de clases Contabilidad.



Título:	Diagrama de clases Gestión de Usuarios.

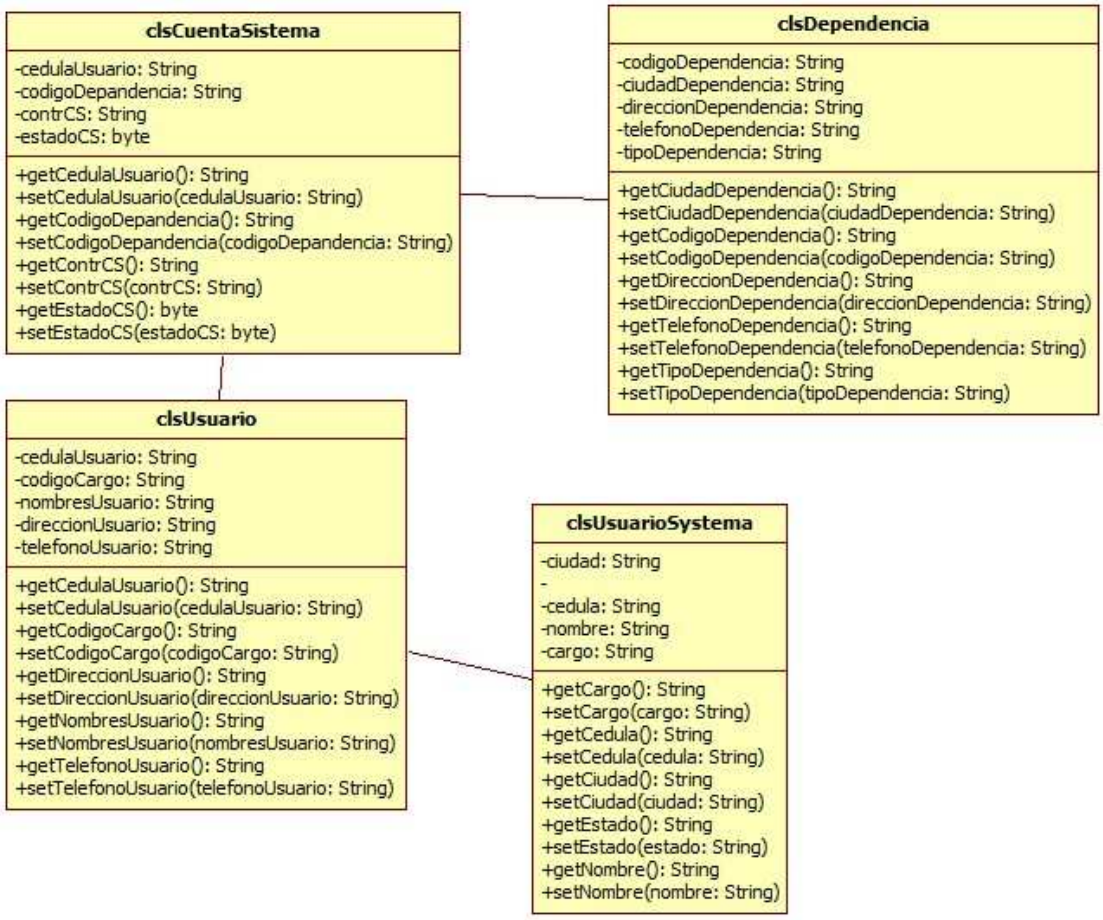
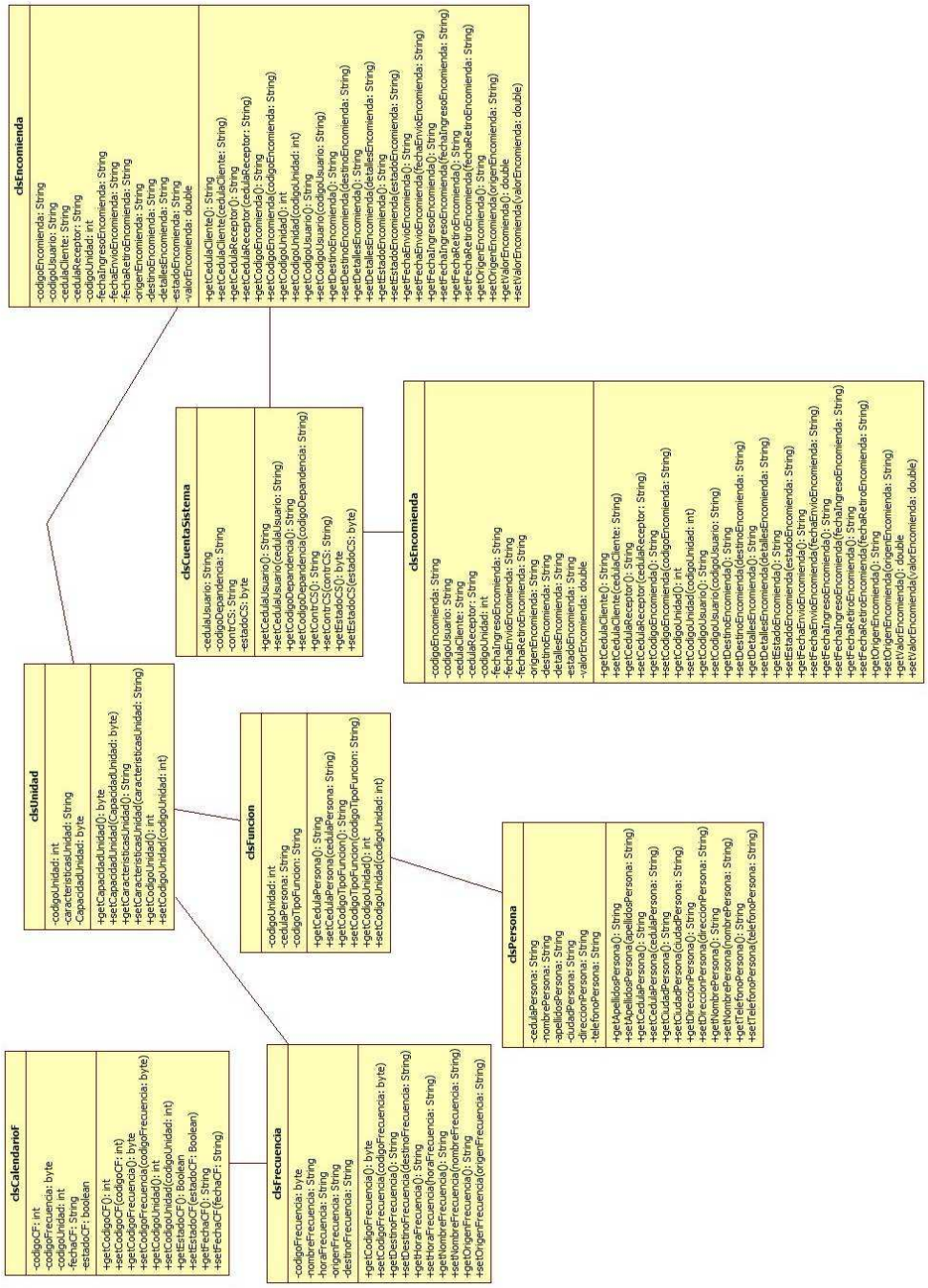


Figura II.9 Diagramas de Clases

Título:	Diagrama de clases General.



2.1.3. Diseño Físico

2.1.3.1. Diagramas de Actividades

Título:	Registro de Venta de Pasaje

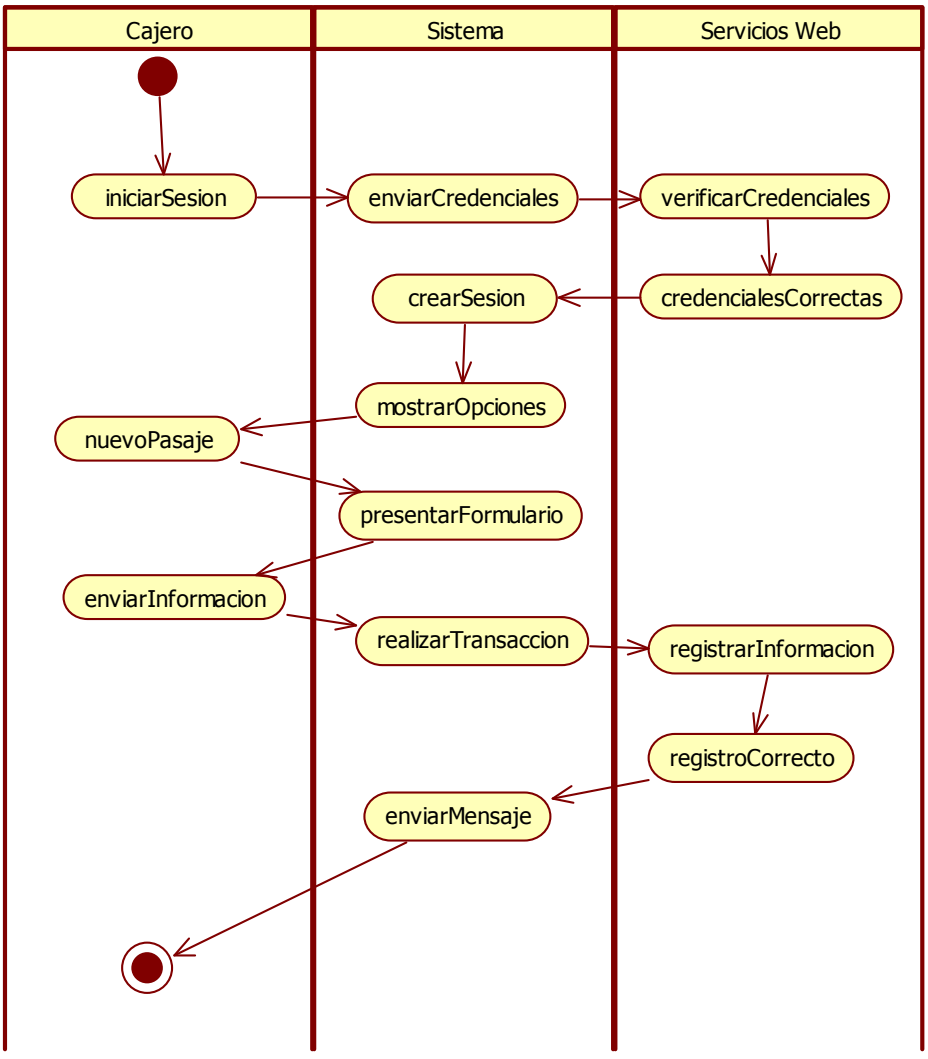


Figura II.10 Diag. Actividades Cajero

Título:	Consultas Gerente

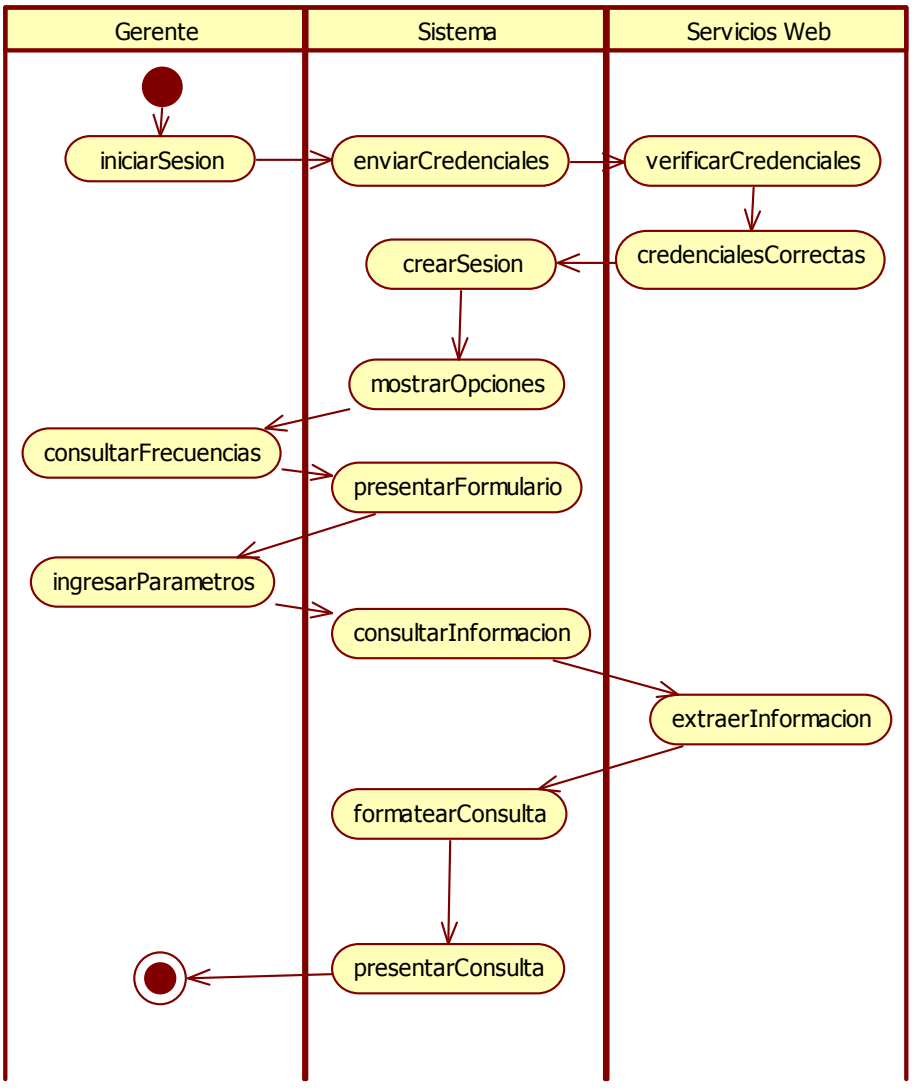
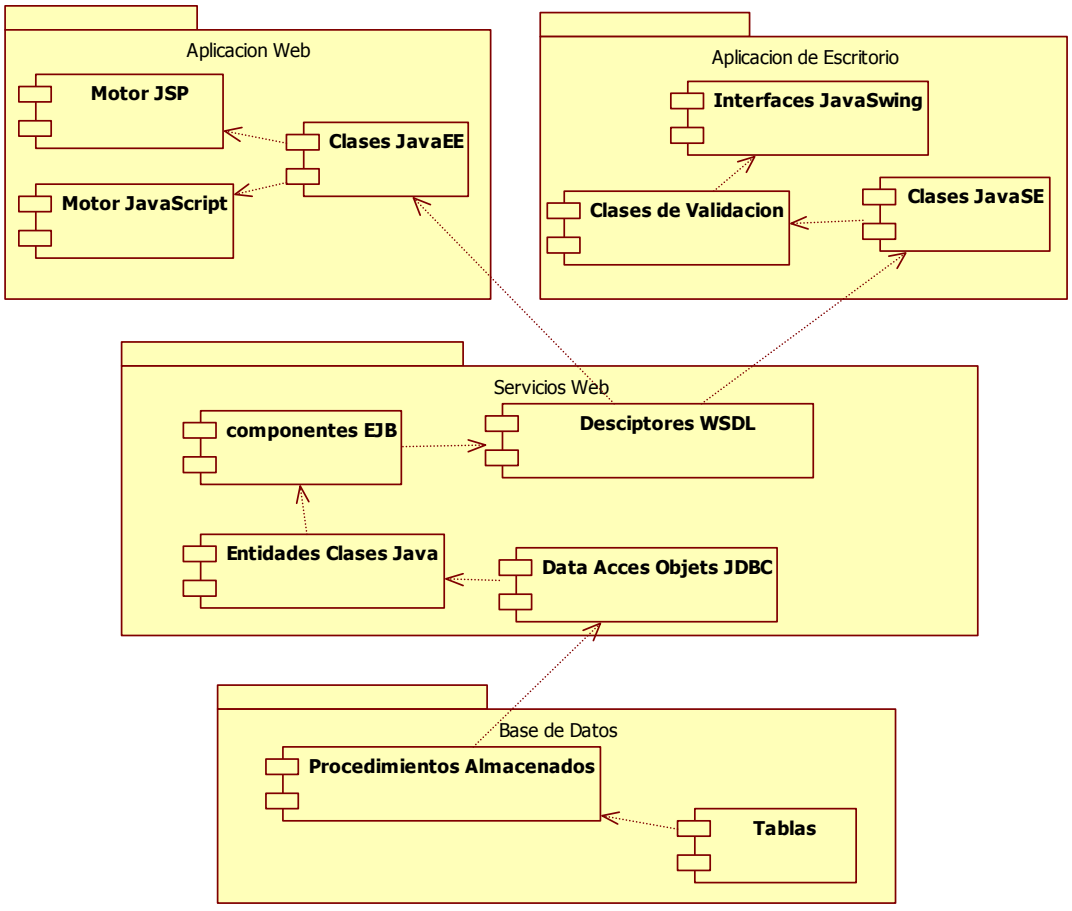


Figura II.11 Diag. Actividades Gerente

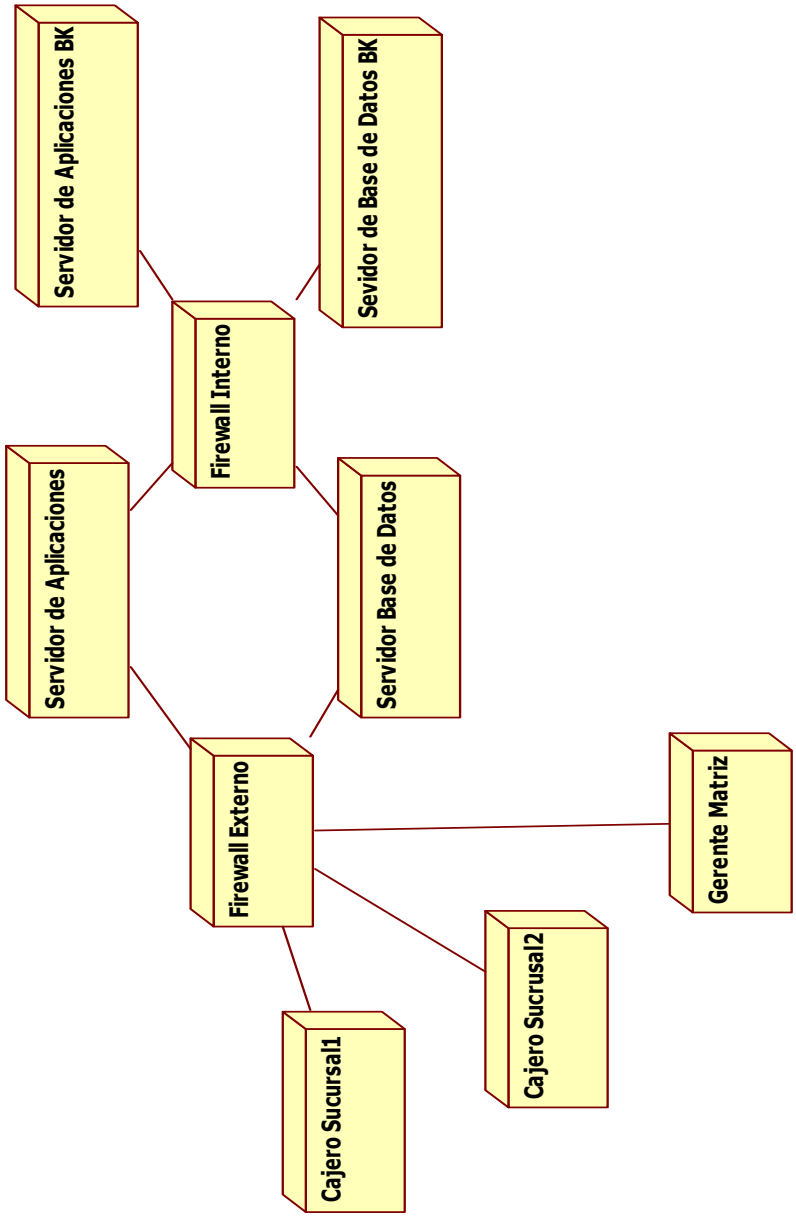
2.1.3.2. Diagrama de Componentes

Título:	Componentes Generales



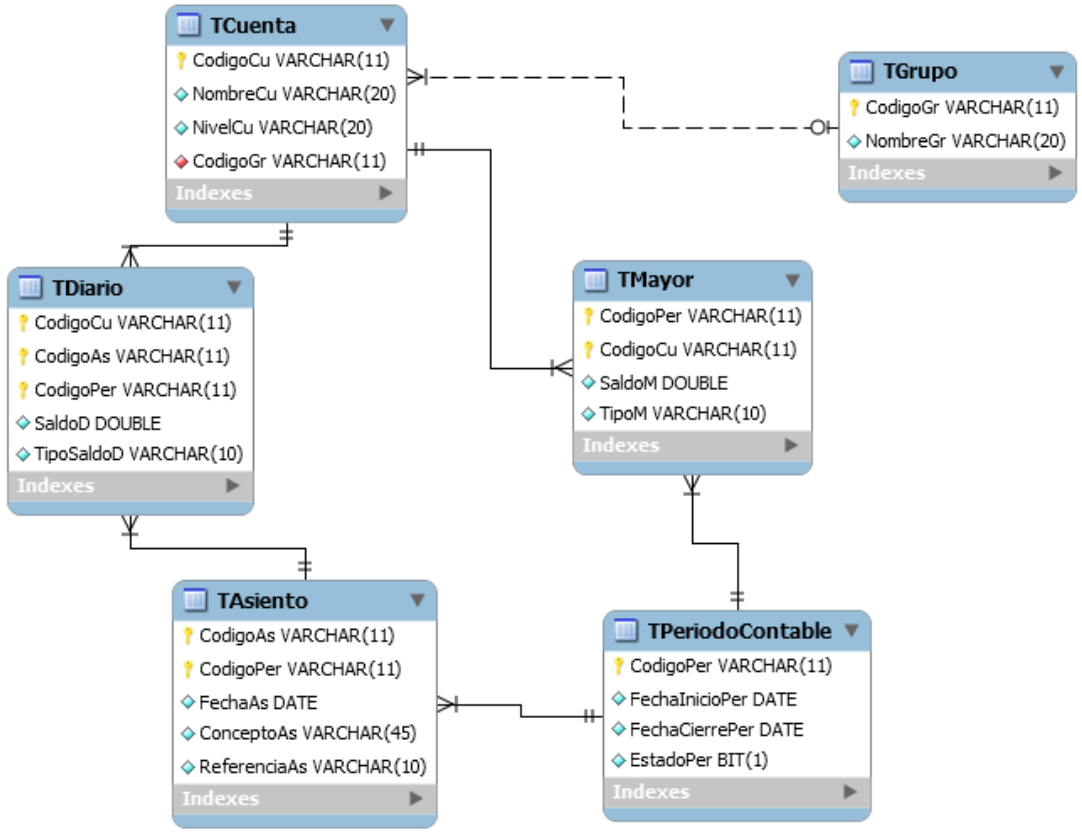
2.1.3.3. Diagrama de Implementación.

Título:	Despliegue General

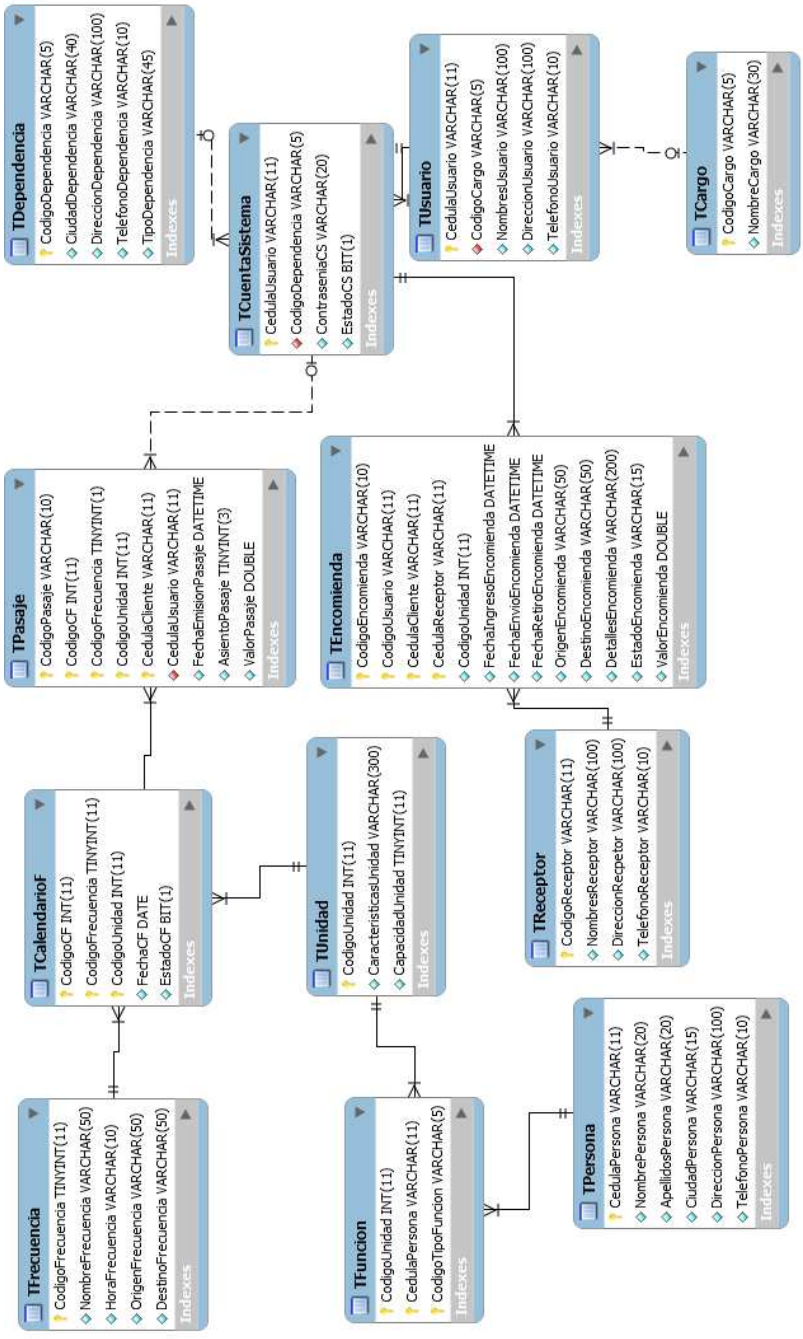


2.1.3.4. Modelo Físico de la Base de Datos

Título:	Base de Datos Contabilidad



Título:	Base de Datos Ventas



Fase 3: Desarrollo

3. Desarrollo

3.1. Nomenclatura y Estándares para el Desarrollo

NOMENCLATURA DE PROGRAMACIÓN

	AMBITO		
	Local o a nivel de procedimiento	**Nivel de modulo o form**	**Global**
Controles			
Classes	cls< NombreClase>	cls< NombreClase>	cls< NombreClase>
Componentes	cmp< NombreComp>	lCmp< NombreComp>	cmp< NombreComp>
Formularios	frm< NombreFormulario>	lFrm< NombreFormulario>	gFrm<NombreFormulario>
Combobox	cmb< NombreCombo>	lCmb< NombreCombo>	gCmb< NombreCombo>
Command	cmd< NombreComando>	l Cmd< NombreComando>	gCmd< NombreComando>
Datagrid	grd<NombreGrid>	lGrd<NombreGrid>	gGrd<NombreGrid>
ListBox	lst< NombreListbox>	lLst< NombreListbox>	gLst<NombreListbox>
buttons	btn< NombreBoton>	lBtn< NombreBoton>	gBtn< NombreBoton>
checkBoxes	chk< NombreCheck>	lChk< NombreCheck>	gChk< NombreCheck>
Textboxes	txt< NombreText>	lTxt< NombreText>	gTxt< NombreText>
Tipos primitivos			
Integer	int< Nombre >	lInt< Nombre >	gInt< Nombre >
Long	lng< Nombre >	lLng< Nombre >	GLng< Nombre >
Bolean	bln< Nombre >	lBln< Nombre >	gBln< Nombre >
Object	obj< Nombre >	lObj< Nombre >	gObj< Nombre >
String	str< Nombre >	lStr< Nombre >	gStr< Nombre >
Double	dbl< Nombre >	lDbl< Nombre >	gDbl< Nombre >
Constantes	C_< NOMBRE>	LC_< NOMBRE >	GC_< NOMBRE >
ws	ws<NombreServicioWeb>	ws<NombreServicioWeb>	ws<NombreServicioWeb>

NOMENCLATURA DE BASE DE DATOS

Estándar	Nomenclatura	Descripción
tbl	tbl<NombreTabla>	Para nombrar a las tablas
txt	txt<NombreCampo>	Para nombrar a los campos de tipo texto
dt	dt<NombreCampo>	Para identificar a los campos de tipo fecha
num	num<NombreCampo>	Identificar datos de tipo numérico
byt	byt<NombreCampo>	Identificar datos de tipo byte
dbl	dbl<NombreCampo>	Identificar datos de tipo double

3.2. Capa de Presentación

3.2.1. Diseño de Interfaces de Usuario

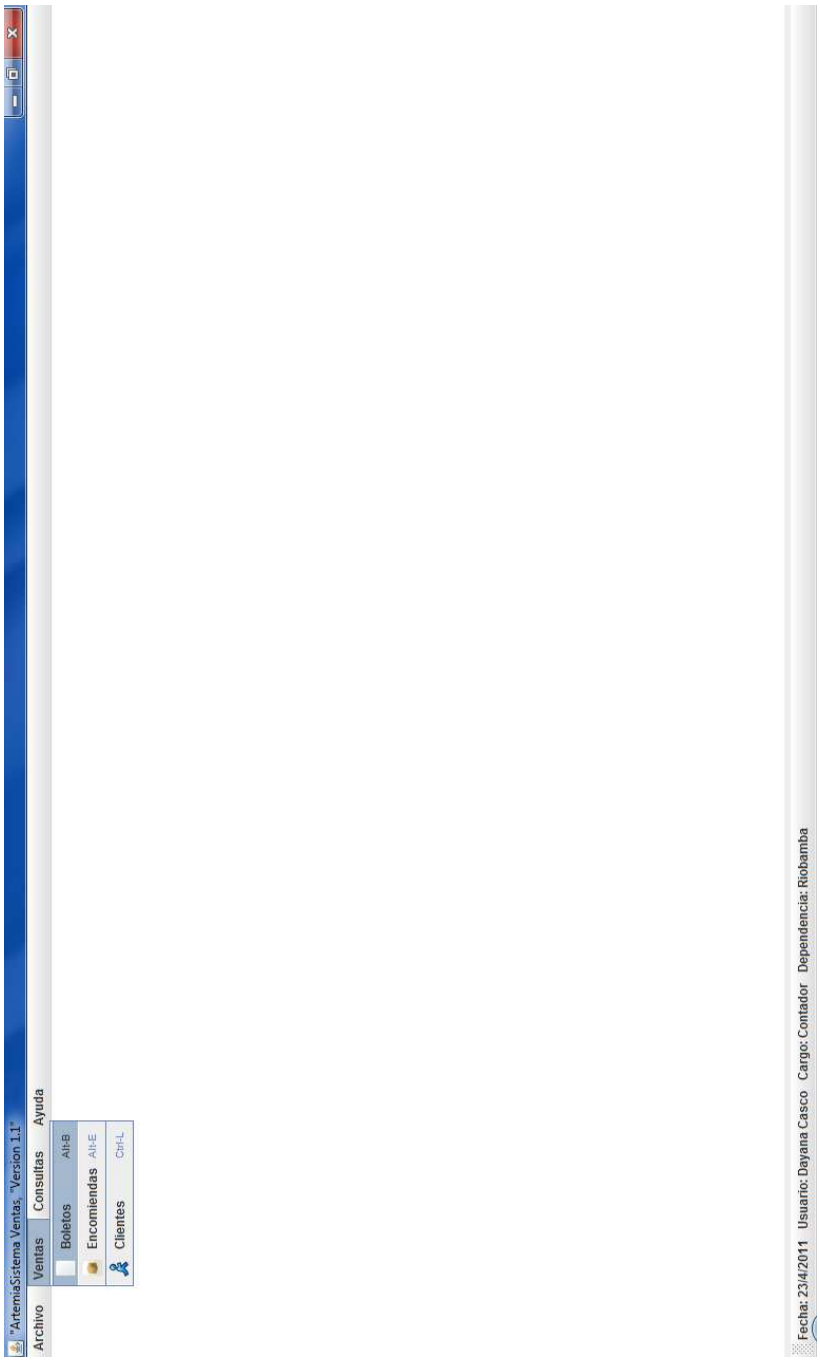
Título:	Pantalla de Inicio de Sesión

Figura III.1 Inicio de Sesión

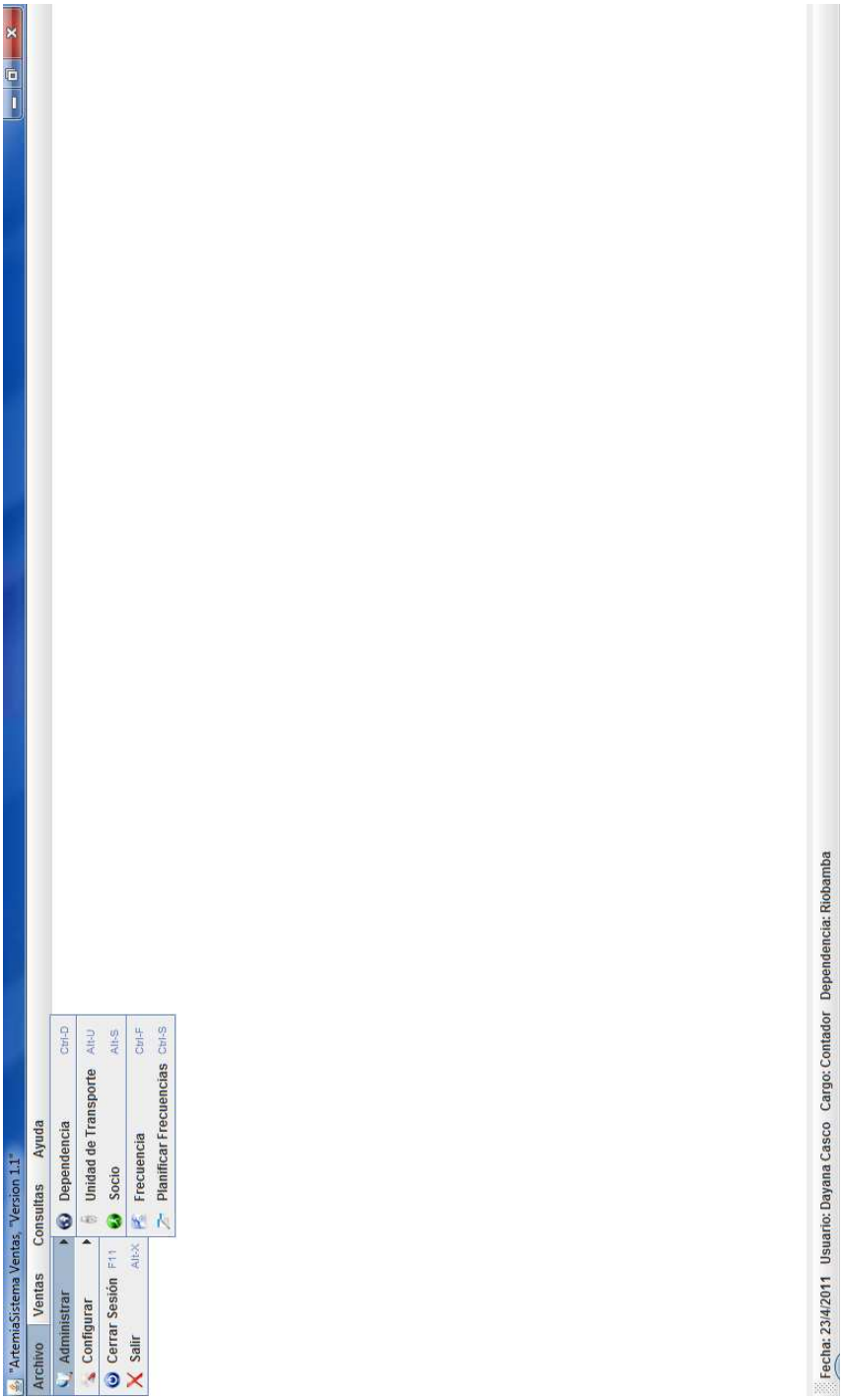
Título:	Pantalla de Venta de Pasajes.

Interfaces de usuario.

Título:	Pantalla Menú Ventas



Título:	Pantalla Menú Archivo



Título:	Pantalla Consultas Gerente vía Web.

Figura III.2 Aplicación Web Gerente

3.3. Capa de Datos

3.3.1. Implementación de la Base de Datos

Para la implementación de la base de datos se utilizó el modelo relacional y como herramienta case a MySQL Workbench, el acceso a la información se brinda mediante procedimientos almacenados.

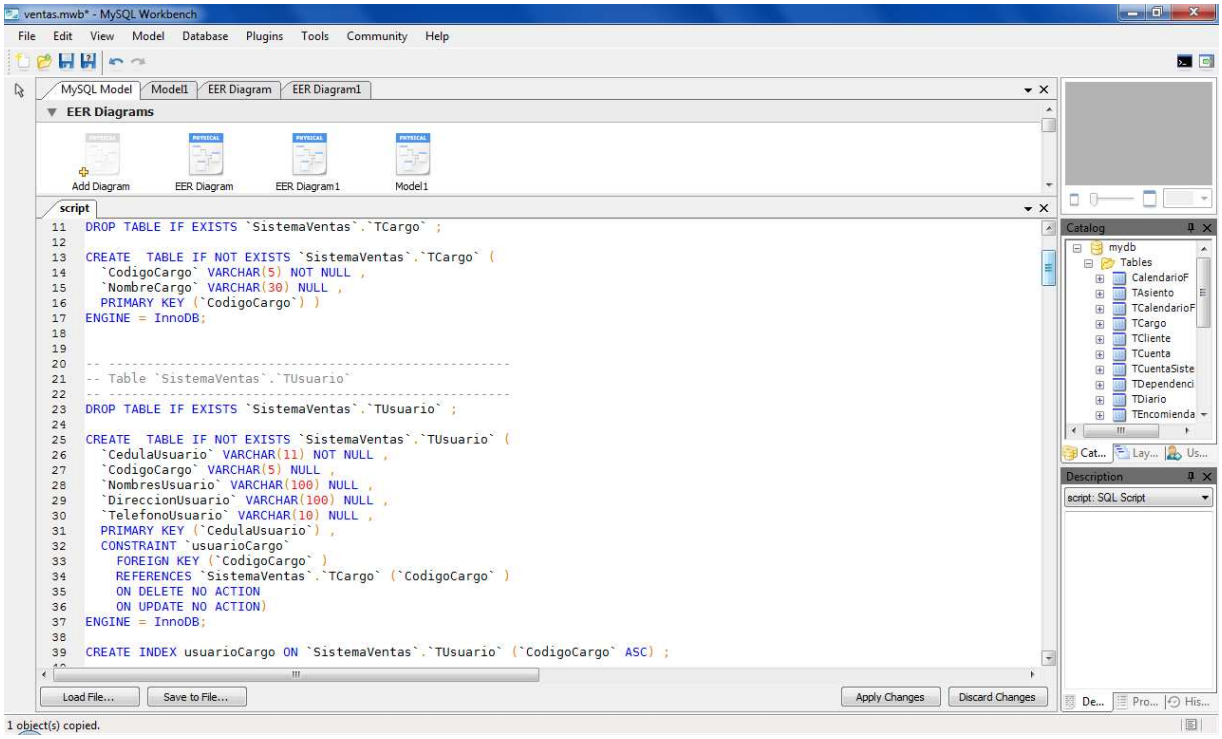


Figura III.3 Implementación de la Base de Datos

3.3.2. Implementación de Acceso a Datos

El acceso a datos se implementó mediante clases JAVA y el API JDBC para la conexión a MySQL, como herramienta se utilizó el IDE de Oracle NetBeans.

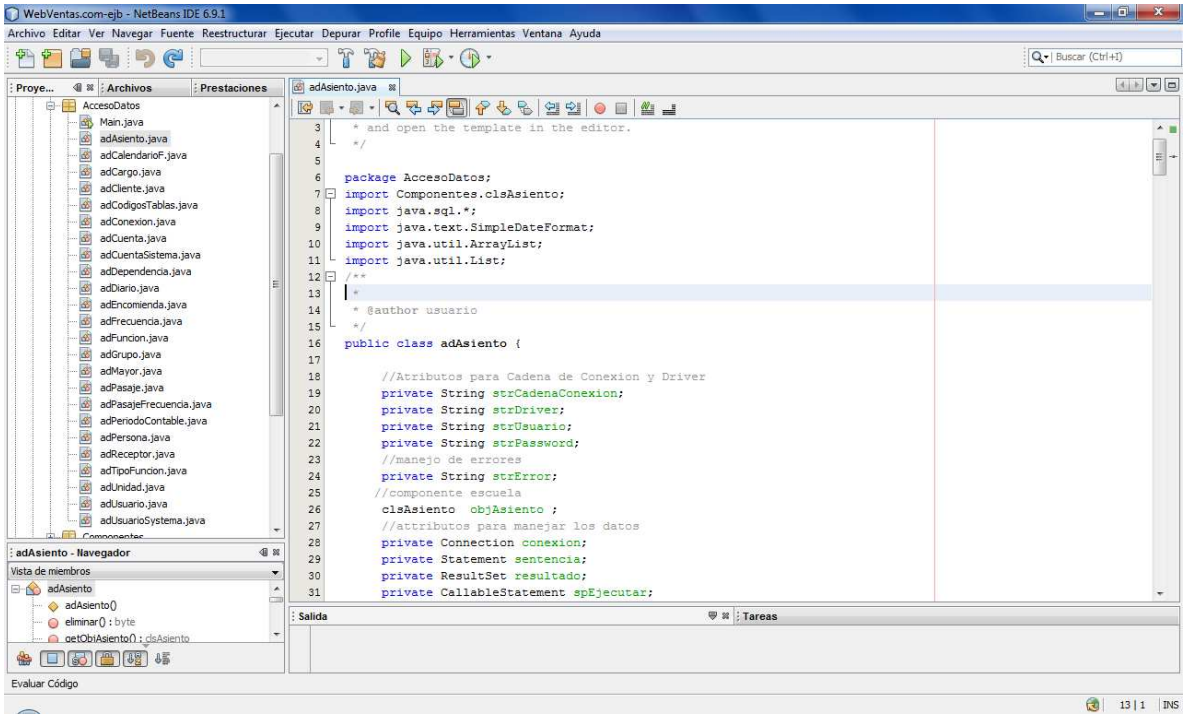


Figura III.4 Implementación del Acceso a Datos

3.4. Capa de Negocios

Aquí se implementan todas las reglas de negocio reflejadas mediante la lógica funcional que hace que el sistema cumpla con los requerimientos necesarios.

3.4.1. Implementación de componentes

Los componentes son clases JAVA con funcionalidades específicas para el manejo de las operaciones en las entidades de la Base de Datos.

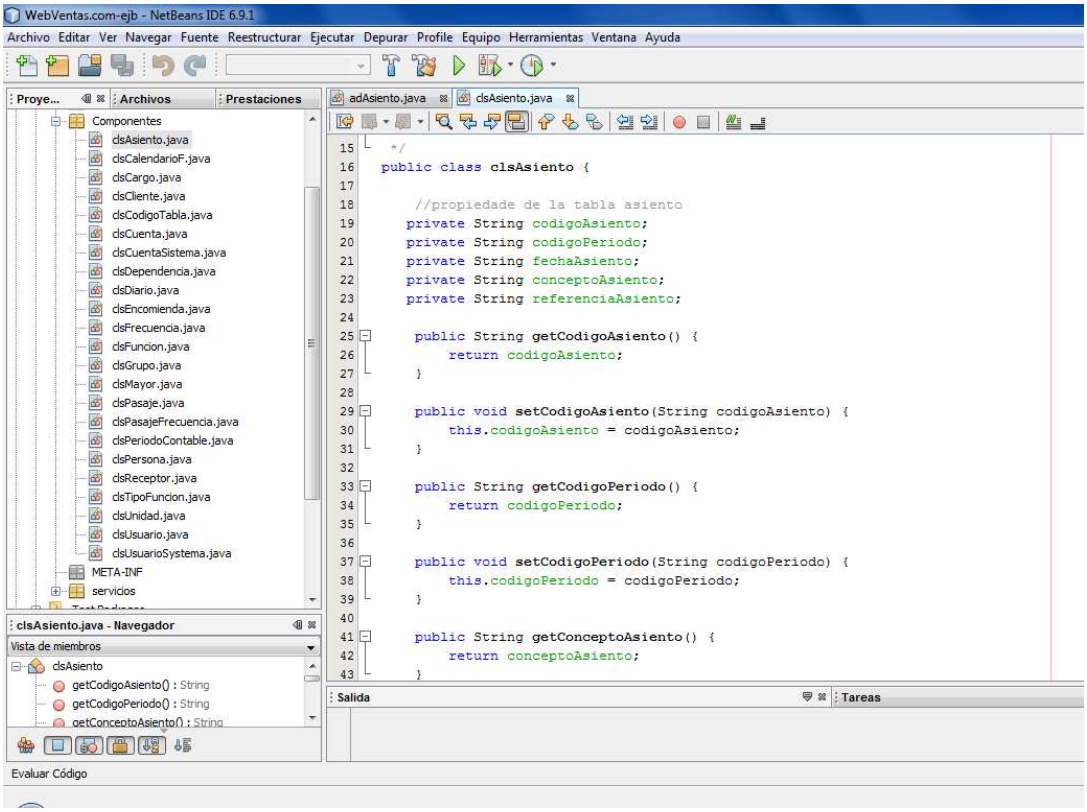


Figura III.5 Implementación de Componentes

3.5. Especificaciones de Seguridad

3.5.1. Diseño de Estrategias de Autorización, Autenticación y Auditoria

3.5.1.1. Autenticación y Autorización

Para el control de acceso a las diferentes funcionalidades del proyecto se ha implementado el manejo de sesiones y la autenticación de los diferentes usuarios.

- ✓ El manejo de sesiones hace que las peticiones hacia el servidor sean atendidas únicamente a los usuarios registrados en la base de datos del sistema y sólo desde los equipos (direcciones IP) desde los cuales se inicie la sesión.

- ✓ Una medida de seguridad adicional se consigue mediante el almacenamiento de la información confidencial de los usuarios en la base de datos del sistema mediante el uso de algoritmos de encriptación.

- ✓ El servidor de Base de datos maneja un único usuario que acepta conexiones y peticiones de transacción remotas. De esta forma se asegura que la información que se entrega sea únicamente al servidor autorizado en el que se encuentren alojados los distintos servicios para el acceso público de los clientes.

3.5.1.2. Comunicación Segura(SSL, IPsec, VPNs)

La comunicación entre las distintas capas del sistema se hace de forma segura, la capa de servicios web maneja el protocolo HTTP seguro utilizando el protocolo de comunicación SSL.

La configuración recomendada para la comunicación entre los distintos servidores es la siguiente:

- ✓ La topología de Red constará de un segmento de Intranet y uno de extranet, separados por un equipo de Firewall. En el segmento de intranet se encontrarán los servidores de bases de datos, el equipo donde se encuentran publicados los servicios web y un servidor de respaldo para el servidor Web, el cual se encontrará funcionando en el segmento de Extranet, al cual se podrán conectar los clientes desde internet. Esta configuración será posible mediante el manejo de Redes Privadas Virtuales.

3.5.1.3. Almacenamiento de información sobre las actividades de los usuarios para posteriores auditorias.

Es importante el registro de las distintas actividades de los usuarios respecto a las funcionalidades brindadas por el sistema, de esta forma se

puede posteriormente controlar y analizar el comportamiento del sistema en horas pico o seguir el rastro de posibles accesos indebidos. Esta forma de bitácora es importante para futuros procesos de mantenimiento y pruebas de la seguridad del sistema.

3.5.1.4. Proveer a los usuarios sólo los privilegios necesarios para cumplir con sus actividades.

El diseño de la solución propuesta siempre mantendrá presente el perfil del usuario que se encuentra utilizando el sistema, cuidando que se le presenten únicamente las opciones necesarias y suficientes para el cumplimiento de sus actividades, cuidando de esta forma la integridad de la información que se almacene y por ende la integridad del sistema.