

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTUDIO DE FACTIBILIDAD DE LOS WEB SERVICES IMPLEMENTADOS CON SOFTWARE LIBRE

T E S I S:

QUE PARA OBTENER EL TÍTULO EN

INGENIERIA EN COMPUTACIÓN

P R E S E N T A N

ARROYO SÁNCHEZ ÁNGEL EDUARDO
MARTÍNEZ HERNÁNDEZ OSCAR



DIRECTOR DE TESIS: ING. ALEJANDRO VELÁZQUEZ MENA

México D.F. 2006

FACULTAD DE INGENIERIA

ESTUDIO DE FACTIBILIDAD DE LOS WEB SERVICES IMPLEMENTADOS CON SOFTWARE LIBRE

T E S I S:

QUE PARA OBTENER EL TITULO EN

INGENIERÍA EN COMPUTACIÓN

P R E S E N T A N

ARROYO SÁNCHEZ ÁNGEL EDUARDO
MARTÍNEZ HERNÁNDEZ OSCAR

DIRECTOR DE TESIS: ING. ALEJANDRO VELÁZQUEZ MENA

MEXICO D .F. 2006

AGRADECIMIENTOS
Ángel Eduardo Arroyo Sánchez.

Señor, sé que has iluminado mi camino con tu amor y sabiduría infinita. Me reconforta el hecho de tu compañía y saber que las cosas tienen un lado más profundo y con sentido. Gracias por brindarme tu luz que me ha impulsado y guiado hacia éste camino.

A mi esposa Jessy:

Tu presencia en éste tiempo me hace entender lo valioso de tu compañía; tu amor, me regocija y me llena completamente. Veo un motivo muy grande en ti y un deseo de dedicarte y compartir todos mis éxitos a una mujer tan tierna y llena de amor como lo eres tú. Gracias mi vida, te amo inmensamente que Dios te bendiga siempre.

A mis padres Doris y Francisco:

Mamita hermosa mi agradecimiento y admiración por entregar y dedicar tu amor, tus cuidados, tu atención, tu empeño apasionado en mi formación, lo cual es sin lugar a duda, un éxito tuyo en el cual ahora lo exalto y lo admiro enormemente. Estoy seguro de sentir tu alegría y orgullo desde mi corazón que es el lugar donde yo te encuentro.
Gracias papá por ser tan grande como mamá al brindarme tu apoyo y estar ahí, eres para mi un amigo donde he podido hallar en tus consejos, además de tu ejemplo, la forma de enfrentar la adversidad. Todo esto ha conformado las bases en las cuales me guiarán en el camino y nunca me harán sentir solo.
Con todo mi cariño dedico este esfuerzo de ustedes y para ustedes. Con orgullo y reconocimiento decirles que siempre los llevo en lo mas interno de mi ser y los quiero infinitamente

U.N.A.M. y profesores

Mi agradecimiento y orgullo por pertenecer a esta gran institución, donde recibí más que una serie de conocimientos, un espacio libre y enriquecedor para la formación no sólo como profesionista sino también como individuo con responsabilidad ética y una visión critica del entorno donde se vive.

A mis Hermanos:

Gracias Paco y Ale por ser parte de los momentos más importantes en mi vida, comparto mi alegría de culminar este proyecto y decirles que no olviden que los quiero mucho reconociendo sus deseos y apoyo en cada instante. Dios los bendiga.

Un Agradecimiento y dedicación especial a:

Mis abuelitos, a un angelito llamado Emiliano y a mis lindas tías: Ivonne, Clarita y Silvia que me brindaron su compañía, apoyo, bendiciones y oraciones cuando más lo necesitaba. Con mucho cariño para todos ustedes.

AGRADECIMIENTOS
Oscar Martínez Hernández.

A mi Abuelo Wilfrido:

Te agradezco todas las enseñanzas que me brindaste, no solo de palabra, si no de actitud y acción, y el apoyo incondicional que siempre ofreciste.

Me considero una persona afortunada por el hecho de haberte conocido y convivido juntos, ya que ahora se que en este planeta aun existen personas con gran sentido de humanidad, humildad, responsabilidad y compromiso.

Espero te sientas orgulloso de los logros que hasta ahora he obtenido, y disculpa si no pongo en practica todos tus enseñanzas, pero en verdad es complicado seguir tu ejemplo.

Siempre te llevo en mi mente y recuerdo por que se que aun sigues aquí, gracias...

A mi hermano Luis Carlos:

Gracias Carlos por todo el apoyo brindado desde la infancia hasta el día de hoy.

Hay cosas inexplicables en la vida, no sabemos por que suceden pero ahí están, estoy seguro que no pude pedirle a la vida un mejor hermano y amigo que tu.

Eres una persona de la cual estoy muy orgulloso por todos los logros que has obtenido y seguramente seguirás obteniendo.

A mi Madre Gloria:

Gracias por la fortaleza que siempre muestras ante cualquier adversidad que se nos presenta, por que gracias a está he formado el carácter y la persistencia para lograr todos mis objetivos.

Quiero que sepas que en cada una de mis metas alcanzadas, siempre participas mamá, directa o indirectamente.

Hay un aprendizaje que practico día a día, y es uno de los más valiosos para mí, es el ser una persona que persigue sus ideales y antepone ante cualquier circunstancia estos, eso lo aprendí de ti, gracias MAMA.

Ahora nos llego el tiempo de disfrutar todo por lo que luchaste desde que éramos pequeños hasta vernos unas personas profesionistas.

A mi padre Jose luis:

Gracias por que se que siempre quisiste lo mejor para nosotros y que siempre estuvimos en tus pensamientos, espero te encuentres bien.

U.N.A.M.

Mi agradecimiento por pertenecer a esta gran institución, que me albergo dentro de sus instalaciones y fue el primer contacto que tuve con el ámbito profesional.

Gracias a los profesores de la facultad, que de manera incondicional me brindaron sus conocimientos, consejos y experiencia profesional.

TESIS

ESTUDIO DE FACTIBILIDAD DE LOS WEB SERVICES IMPLEMENTADOS CON SOFTWARE LIBRE

INDICE

INDICE DE FIGURAS
INDICE DE TABLAS

INTRODUCCIÓN……..... 7

I. CONCEPTOS, PRINCIPIOS Y TEMAS RELACIONADOS CON LOS WEBSERVICES…. 12

1.1 ANTECEDENTE DE LOS SISTEMAS DISTRIBUIDOS……..... 12

1.2 INTRODUCCIÓN A LOS WEB SERVICES……...… 13

1.3 INTRODUCCIÓN SOAP……..... 16

1.3.1. INTERCAMBIO DE MENSAJES EN SOAP……..... 17
1.3.2. ASUNTOS AVANZADOS DE LA COMUNICACIÓN SOAP……...…….…..... 20
1.3.3 CODIFICACION……..…… 21
1.3.4. RESUMEN SOAP……..……...…….…..... 22

1.4 INTRODUCCIÓN A WSDL……..... 23

1.4.1. DOCUMENTO WSDL……..……..... 23
1.4.2. ANATOMIA DE LOS DOCUMENTOS WSDL……..... 23
1.4.3. ARCHIVOS FISICOS……..……..... 25
1.4.4. DEFINICIÓN WSDL ……..……..... 26
1.4.5. DEFINICIÓN DEL SERVICIO……..... 26
1.4.6. RESUMEN WSDL……...…..... 26

1.5 INTRODUCCION A UDDI……..……. 27

1.5.1. VISIÓN GENERAL UDDI……..……..... 27
1.5.3. ESTRUCTURA DEL REGISTRO UDDI ……..……..... 28
1.5.4. INTERACCIONES CON UDDI……..……..... 30
1.5.6. ENCONTRANDO INFORMACION DESDE LAS OPCIONES DE BUSQUEDA DE UDDI ……..…… 32
1.5.7. PUBLICANDO INFORMACIÓN DEL NEGOCIO UTILIZANDO LOS REGISTROS UDDI ……..…… 34
1.5.8. ESCRIBIENDO CLIENTES UDDI……..……..... 36
1.5.9. REGISTROS PRIVADOS UDDI……..……..... 40
1.5.10. POSIBLES ESCENARIOS PARA REGISTROS PRIVADOS UDDI……..……..... 41
1.5.11. BENEFICIOS DE REGISTROS PRIVADOS UDDI……..……..... 42
1.5.12. RESUMEN UDDI……..…….. 43

II UTILIZACIÓN DE SOFTWARE COMERCIAL (WEBSPHERE)…….….. 44

III. IMPLEMENTACION DE UN WEBSERVICE UTILIZANDO JBOSS.

3.1 REQUERIMIENTOS Y SOFTWARE A UTILIZAR……..... 46
3.2 APLICACIÓN DE SERVIDOR JBOSS…… 47
3.3 LA ARQUITECTURA J2EE……..… 47
3.3.1. EJB……..... 49
3.4 ASPECTOS DE LA APLICACIÓN DUKES BANK……..... 51
3.5 DESPLEGANDO WEBSERVICE DE LA APLICACIÓN DUKESBANK……. 52
3.6 EJEMPLO DE WEBSERVICE UTILIZANDO JBOSS WS TELLERSERVICE……..... 60

IV ANÁLISIS DE FACTIBILIDAD DE APLICACIÓN DE UN WEBSERVICE…….. 66

4.1 PLANEACIÓN ESTRATEGICA DE UN WEBSERVICE…….... 66
4.2 METAS Y RETOS DE LOS WEB SERVICES…… 70
4.3 LA PROPUESTA DEL VALOR DE LOS WEB SERVICES…….…….... 74
4.4 OPORTUNIDADES PARA IMPLEMENTAR WEB SERVICE……. 75
4.5 ALTERNATIVAS DE SOFWARE PARA EL PROYECTO WEB SERVICES FRENTE A J2EE Y SOFWARE LIBRE ……..…… 75
4.5.1 IMPLEMENTACIÓN DE UN WEB SERVICE UTILIZANDO PLATAFORMA J2EE..... 76
4.5.2. OPCIONES A J2EE DONDE UNA APLICACIÓN PUEDE SOPORTAR LA IMPLEMENTACIÓN WEB SERVICE…… 77

4.6 IMPLEMENTACIONES CON SOFTWARE DE LICENCIA LIBRE (OPENSOURCE)……..... 80
4.7 ESQUEMA WEB SERVICE COMO UNA APLICACIÓN DE NEGOCIO……. 83
4.8 WEB SERVICES SEGUROS…….. 86
4.9 RESUMEN DE VENTAJAS Y DESVENTAJAS DE LOS WEB SERVICES UTILIZANDO SOFWARE LIBRE……..……..88

CONCLUSIONES …… 91

ANEXOS O APENDICES ……..…….....99

ANEXO 1 ……..…….…….... 99

(A) MENSAJE SOAP EMBEBIDO EN UNA SOLICITUD HTTP 99
(B) MENSAJE SOAP EMBEBIDO EN UNA RESPUESTA HTTP 99
(C) MODELO DE COMPONENTES SOAP DE ALTO-NIVEL 100

ANEXO 2…….. 101

(A)EJEMPLO SIMPLE DE UN WSDL PARTE 1 ……..... 101
(B)EJEMPLO SIMPLE DE UN WSDL PARTE 2 ……..... 101
©WSDL USA LOS XML NAMESPACE (NOMBRES DE ESPACIO) LISTADOS EN LA TABLA . 102
(D) TIPO DE DEFINICIÓN EN EL DOCUMENTO WSDL ...……..... 103

ANEXO 3 REGISTROS UDDI ……... 104

ANEXO 4 API BASE PARA WEB SERVICES EN LA PLATAFORMA JAVA (JAX-RPC) 105

ANEXO 5 FUNCIONAMINETO DE LA APLICACIÓN DUKES BANK …….……... 108

ANEXO 6 SUBSISTEMAS DE AXIS ……..…. 112

ANEXO 7. INSTALACIÓN DE LA APLICACIÓN DUKE´S BANK EN EL SERVIDOR DE APLICACIÓN JBOSS ……..…… …. 113

ANEXO 8. IMPLEMENTACION DEL SESSION BEAN TELLER COMO WEBSERVICE EN JBOSS …..……..……..… 122

ANEXO 9. IMPLEMENTACIÓN DEL WEB SERVICE TELLER CORRIENDO EN EL SERVIDOR JBOSS 4.0.1 ……...124

GLOSARIO DE TÉRMINOS……..……...……..... 134

BIBLIOGRAFÍA 136

INDICE DE FIGURAS

1. CONCEPTOS DE LOS WEB SERVICES

Componentes WebServices 14
Esquema Implementación WebServices 16
Interacción SOAP entre cliente y servidor..... 20
Componentes de implementación del servidor apache SOAP..... 22
Elementos WSDL y sus relaciones..... 24
Estructura de un documento WSDL como uno, dos o tres archivos..... 25
Relación de Entidades del modelo de datos UDDI y la liga WSDL.....29
Interacción con registros UDDI..... 30
Opciones para búsqueda de los registros UDDI..... 32
Explorando detalles del servicio publicado.....33
Expandiendo entidades de negocio para exploraciones existentes o definir nuevos servicios..... 35
Descripción de artículos publicados..... 35
Método find_bussiness 35
Método find_services 37
Método find_tModel 37
Método find:binding 39
Ejemplo de como registrar información en servidor UDDI 40

II. IMPLEMENTACION DE UN WEBSERVICE UTILIZANDO JBOSS.

Principales componentes de una arquitectura J2EE 48
Esquema componentes J2EE y descriptores de despliegue.....49
Esquema de componentes de la aplicación Dukes Bank 52
Inicio del servidor Jboss versión 3.2.1 en WindowsXP.....54
Administración Hipersonic..... 55
Despliegue de la Aplicación en JBOSS.....55
Descripción delWebService.....56
Despliegue del WebService Controller archivo ebank,wsr..... 57
Localización del Web Service a través de Visual Basic Studio .Net.....57
Web Services disponibles en JBOSS-NET.....58
Página Controller.wsdl (describe el WebService).....58
Visual Basic.Net Inicio del proyecto cliente.....59
Grafica de la implementación Web Service Teller en Jboss.....60
Consola Web de JBoss-4.0.1.....61
Despliegue de la aplicación en JBoss-4.0.1 (JBossDukesbank.ear).....62
Despligue de los WebServices Disponibles por el servidor vista en el explorador.....62
Página WSDL :Archivo Teller.wsdl.....62
Corrida programa cliente java del WebService Teller.....63
Comportamiento del servidor JBoss-4.0.1.....63
Localización del servicio a través Visual Studio.....64
Navegador en Visual Studio desplegando la página Teller.wsdl.....64
TPCMonitor: Herramienta de Axis permite exponer el comportamiento de los mensajes SOAP que componen al WebService Teller 65

III. ANÁLISIS DE FACTIBILIDAD DE APLICACIÓN DE UN WEBSERVICE

Prioridades de factibilidad en la organización..... 69

Tendencia de tecnología orientada a los sitemas distribuidos..... 72
Gráfica de Relación de alcance de los WebServices..... 74
Servidores de aplicación con licencia libre que soportan WebSerrvices basados en J2EE..... 81
Herramientas IDE..... .82
Integración posible a través de un Web Service..... 83

ANEXO1

Mensaje SOAP embebido en una solicitud HTTP..... 99
Mensaje SOAP embebido en una respuesta HTTP..... 99
Modelo de componentes SOAP de alto-nivel 100

ANEXO 2

(a) Ejemplo simple de un WSDL parte 1..... 101
(b) Ejemplo simple de un WSDL parte 2..... 101
Tipo de definición en el documento WSDL.....103

ANEXO 3

Tabla registros UDDI.....104

ANEXO 4

Estructura JAX-RPC.....106

ANEXO 5

Página principal de la aplicación: Autenticación.....108
Menú de transacciones de la aplicación.....108
Transferencia de cuentas.....109
Saldo de cuentas.....109
Cajero automático ATM.....110
Estado de Cuenta.....110
Resultado de transacción.....111

ANEXO 6

Composición de Apache Axis.....112

ANEXO 7

Archivo JBoss-build.xml.....114

ANEXO 8

Programa cliente en lenguaje java.....122

ANEXO 9

Teller Bean 130
Cliente Visual Basic :NET 131
Abrir Nuevo Proyecto Visual Studio.Net.....132
Entorno Visual Basic Studio .Net WebServices 132
Localizaciòn WebServices Disponibles 132
Creación de la forma WebServiceTeller.....133

INDICE DE TABLAS

CAPITULO 1

CONCEPTOS DE LOS WEB SERVICES
Nombre de espacio SOAP.....16

CAPITULO 3

IMPLEMENTACION DE UN WEBSERVICE U.TILIZANDO JBOSS
Descriptores de despliegue 45
Tipos de enterprise Java Beans 49

CAPITULO 4

ANÁLISIS DE FACTIBILIDAD DE APLICACIÓN DE UN WEBSERVICE

Planteamiento de un Modelo de Planeación Estratégica de un Web Services en la empresa...66
Metas en los negocios de alto nivel 69
Retos del entorno actual 69
Tabla comparativa de tecnologías orientadas a los sistemas distribuidos.....72
Servidores de aplicaciones certificados J2EE.....75

ESTUDIO DE FACTIBILIDAD DE LOS WEB SERVICES IMPLEMENTADOS CON SOFTWARE LIBRE

INTRODUCCIÓN

El presente proyecto aborda el tema de una reciente aplicación tecnológica llamada Web Services enfocada a la aplicación de los sistemas distribuidos, a comunicar sistemas de forma efectiva, eliminando el problema de interoperabilidad como se tenía en tecnologías anteriores como lo era la tecnología CORBA (Common Object request Architecture) o DCOM de Windows, los cuales no tuvieron un nivel de aceptación total, debido a la limitaciones en cuanto a la interoperabilidad de las aplicaciones y cuyos objetivos también han sido orientados a la distribución de los sistemas y aplicaciones propios.

Hay que mencionar, que generalmente las aplicaciones de software eran entidades monolíticas, ubicadas en un servidor central o mainframe por ejemplo. Contenían la interfaz de usuario, los datos y la lógica de negocio para manipularlos: Desde el punto de vista de los programadores esto forma de trabajo resulta complicada al momento de reutilizar los componentes de software. Con base a estos, surgieron nuevas técnicas de diseño e implementación, hasta llegar a la orientación de objetos a lo que hoy se conoce como **Arquitectura Basada en Componentes** .Está arquitectura ataca el problema del código mezclado definiendo distintas capas de funcionalidad , así se podrían definir tres capas: interfaz de usuario, capa lógica y/o capa de negocio y una capa de acceso a datos.

La Arquitectura Basada en Componentes permite construir aplicaciones más fáciles de mantener y de escalar, dado que cada capa encapsula su funcionalidad, y cualquier cambio introducido queda contenido en esa capa. y podrá ser reutilizado de forma transparente . la escalabilidad es posible porque las distintas capas pueden colocarse en servidores diferentes.

Sin embargo, aun con esta arquitectura se presentan unos cuantos problemas, en general, resulta difícil utilizar desde un lenguaje de programación , componentes implementados en otro lenguaje (cambian los tipos de datos, la forma de pasaje de parámetros). Salvando los problemas de interoperabilidad entre lenguajes, se tiene un problema mayor: compartir componentes entre plataformas heterogéneas.(por ejemplo llamar un componente COM escrito en Visual Basic desde una aplicación JAVA). La solución aparente es escribir código de integración en ambas plataformas, de manera que se permita el intercambio de información. Esto al parecer no es una idea apropiada: un cambio en cualquiera de las aplicaciones requeriría modificar el código de integración , con lo cual el mantenimiento general del sistema se tornaría muy difícil.

La Arquitectura Orientada Servicios o de sistemas distribuidos, la cual utiliza una herramienta tecnológica como es Web Services, puede brindar una solución a los componentes, al proponer una estructura en la que las aplicaciones se construyen mediante la conexión de servicios poco relacionados entre si. Un servicio, en términos generales es un componente de software cuya funcionalidad se separa en capas y que posee interfaces comunes que sirven para acceder a su funcionalidad. Esas interfaces establecen un contrato entre el servicio y el cliente o consumidor del servicio, y están definidas en un lenguaje estándar, independiente de la plataforma.

De está forma.técnicamente hablando, un Web Services puede definirse como una arquitectura de software compuesta de servicios pobremente acoplados, descriptos por interfase independientes de la plataforma, y que pueden ser descubiertos e invocados de manera dinámica.

Los Web Services pretenden jugar un papel importante en la arquitectura orientada a servicios o sistemas distribuidos. Esto es porque están construidos sobre protocolos estándar, bien conocidos e independientes de la plataforma. Estos son HTTP, XML,UDDIWSDL Y SOAP, los cuales se descirbien en detalle en el transcurso del primer capitulo así como su funcionalidad..

La combinación de todos ellos pretende cubrir los requerimientos de los sistemas distribuidos. Por ejemplo, la orientación de servicios exige descubrimiento e invocación dinámica. La interoperabilidad que brindan los Web Services a través de HTTP como protocolo de transporte de los mensajes.

Por tanto , la idea es partir del conocimiento previo de cual es el significado de un Web Service, como surge, que empresas han iniciado el proyecto, cuales son los alcances y el futuro mismo de esta tecnología, es decir, hacia donde pretende tener presencia. Así como tomar en cuenta cuales son las aplicaciones que se derivan de ésta, descubrir el esquema y el entorno donde puede ser requerido y quienes pueden hacer uso de esta misma así como los beneficios que se obtienen y sus desventajas.

Nombrar que grupos de desarrollo o empresas de software tanto comercial como libre proporcionan el soporte y las herramientas adecuadas y necesarias para iniciar, difundir, proveer, asesorar un desarrollo de un proyecto Web Service,:Definir alternativas de sofware, herramientas,plataformas, donde sea posible justificar el uso de esta aplicación tecnologica de acuerdo a la situación real de una empresa con relación a su aplicación enfocada al negocio y mejora de procesos.

La finalidad y el propósito de este estudio es enfocarse en un aspecto más que técnico, un tanto a la utilidad beneficios y factibilidad para las empresas y cuales pueden ser los alcances vistos desde un aspecto del negocio, de desarrollo, costo, productividad, objetivos, desempeño, etc. Descubrir si es realmente una opción tecnológica que va a cubrir, mejorar o cambiar totalmente un entorno, y si esta pensando primordialmente como una ayuda a la empresa, en sus procesos tanto internos como externos como pueden ser interactuar con proveedores, y clientes en sus aplicaciones o sistemas orientados a negocios, o bien, desde el pubnto de vista de programación la reutilizaación de componentes que involucren la comunicación entre ellos sin importar plataformas o arquitecturas.

Adentrando en el tema, en una aplicación donde el modelo actual de negocio electrónico no facilita la integración de las aplicaciones de Internet con el resto de los sistemas o software de otras empresas, ya sea por falta de estandarización en las aplicaciones de los clientes o compatibilidad con la tecnología que cada empresa maneja, ya que requiere mayor flexibilidad, eficiencia y eficacia en el momento de compartir datos, de realizar transacciones y lograr una localización mas efectiva de los sistemas de otras empresas.

Así también . XML es la base con que se conforma un Web Service, y a partir de esta surge un estándar capaz de manipular información y comunicarse de una forma más efectiva entre los sistemas de información que hacen uso de datos para diferentes propósitos.

Como cualquier otra herramienta, arquitectura, metodología, etc. relacionado con proyectos, no solo informático, sino de cualquier índole, el elemento inicial e indispensable es la detección de una necesidad en un determinado momento y el posterior análisis de la misma. En este análisis, donde se definen las necesidades que dan origen a la solución, se deben medir las posibilidades inmediatas y futuras de utilizar XML. Pero su utilización puede variar drásticamente según el escenario en el cual se quiera implementar la solución.

Un ejemplo clásico y muy importante es el uso de XML en la integración de sistemas de información heterogéneos (diferentes). En la actualidad el mercado está inundado de aplicaciones específicas y/o verticales que junto a la existencia de aplicaciones generales y/o horizontales, lleva a que muchas veces se deban integrar aplicaciones desarrolladas sobre plataformas, modelos de datos y lenguajes distintos. Así, habitualmente nos encontramos tres opciones cuando se plantea este tipo de problemas:

1. Mantener las aplicaciones que funcionan correctamente e integrar con XML.
2. Cambiar todos los sistemas para conseguir una integración “de fábrica”.
3. No integrar manteniendo las aplicaciones independientes.

Es lógico que la decisión tomada deba sustentarse tecnológicamente y en relación Coste - Beneficio.

La tercera opción, es decir, la existencia de aplicaciones no integradas es problemática, debido a las ineficiencias que se genera en los procesos de la empresa, por lo que esa opción se debe descartar aunque se encuentra en la realidad más veces de lo que sería aconsejable.

La segunda opción, es decir, cambiar todos los sistemas tiene un impacto muy importante en cuanto a costes y a constantes cambios en empresas por lo que muchas veces también es desestimada.

Frente a las otras dos opciones, la utilización de XML permite desarrollar una solución integradora para que puedan comunicarse entre sí los sistemas que están probados y funcionando correctamente, otorgando la ventaja de lograr la mejor integración, con un coste contenido y con las ventajas de lograr resultados en corto plazo.
Por otro lado, hoy en día diversas herramientas de desarrollo de aplicaciones, algunas basadas en tecnologías .NET (por parte de Microsoft) y otras en J2EE (Java 2 Enterprise Edition , Compañía Sun Microsystems).La mayor parte de las empresas utilizan software comercial, debido a su potencialidad, su diversidad en soporte, documentación, garantía etc. Lo anterior aporta a la empresa una confianza de que se logrará un mejor beneficio, que no arriesgue el negocio, aunque para algunas empresas eso signifique un costo alto por el uso de licencias.
Por ello se busca diversificar las alternativas para el desarrollo de los Web Services capaces de brindar la seguridad, robustez, y sobretodo enfocarse en la economía del negocio debido a todo lo que implica.
El presente trabajo pretende establecer, comprobar, evaluar el funcionamiento que se puede alcanzar utilizando herramientas o bien software que utiliza licencia libre.
El proyecto encaminado a evaluar el rendimiento que provee el software libre para la implementación y el desarrollo de una aplicación que trabaje bajo una tecnología Web Services.
El software libre provee una alternativa de desarrollo e implementación, ya que se mejora continuamente mejora y el código esta siempre abierto y disponible para la abierta aportación de la gente que lo utilize.

Aspectos básicos para el desarrollo viable de un Web Services:
Los Web Services cuentan con:

- Descripción del servicio(Service Description)
- Implementación del Servicio (Service Implementation)
- Servicio de Publicación , Descubrimiento y Unión(Publishing,Discovery and Binding)
- Servicio de Invocación y de Ejecución(Service Invocation and Execution)

Dentro del primer capitulo se detalla las características de los puntos arriba mencionados con la intención de brindar una idea de que existen en la actualidad dos plataformas de desarrollo y tener presente de que pueden ser opciones para implementar un Web Service. Pero en el proyecto es importante resaltar el uso de J2EE por diversas razones, las principales son que se hace el uso del lenguaje java , teniendo las característica de ser portable y que existe una gran gama de servidores de aplicación (comerciales y libres) en el mercado que soportan la especificación J2EE, esto brinda flexibilidad para los procesos de desarrollo, ya que la intención del proyecto es tener trabajando software libre, capaz de mostrar que es posible implementar esta aplicación tecnológica a manos de grupos de desarrollo que trabajan con licencia libre.

Con lo anterior, el reto es iniciar un estudio capaz de descubrir y estudiar una implementación de un Web Service utilizando un servidor de aplicación basado en J2EE(JBOSS), herramientas de software que contienen el protocolo SOAP (AXIS) basado en lenguaje XML, un sistema operativo como Linux y finalmente tener un ejemplo de una aplicación de un sistemas basado en tecnología J2EE, donde sean expuestos sus módulos como un Web Service.

Con base a lo anterior, analizar la instalación, configuración y comportamientos de los elementos antes mencionados para del desarrollo de un Web Service.
Determinar si es factible implementar éste esquema de comunicación estandarizada entre componentes de sofware con herramientas de licencia libre, orientados a brindar una aplicación al negocio electrónico como es la invocación y descubrimientos de servicios donde las aplicaciones o sistemas de sofware jugarán un papel primordial para poder interactuar una con otra así como lo se realiza entre personas a través de internet.

Así también, considerar puntos criticos que interviene en la implementación de un Web Service, si realmente el utilizar esta tecnologia requiere o demanda otros componentes, o si estos componentes presentan dificultades para la puesta en marcha , que provecho pueden obtener las aplicaciones de una empresa, para quiénes y para cuáles situaciones ésta orientado y dirigido. En resumen, con este ejemplo de prueba se justificará la viabilidad, las dimensiones y una idea más profundaa hacia donde se ubica la comunicación abierta entre componentes de sofware utilizando software libre. Por tanto su futuro se ve orientado al negocio

electrónicoi y el impacto en la intervención de los procesos de desarrollo de una empresa o un grupo de ingenirá de sofware.

CAPITULO I
Conceptos de los Web Services

El capitulo presenta una definición mas extensa de un WebService, cubriendo detalles en cuanto a sus componenets y funcionalidad, describir conceptos de estándares como SOAP, UDDI, WSDL con enfoque un tanto técnico, pero con la intención de envolver una idea del comportamieto y funciones principales de estos conceptos y su relación entre ellos.

Existe una reseña de antecedentes sobre la evolución que ha tenido a través de los años la idea de computadora distribuida, es decir, la capcidad de interoperar cualquier tipo de sistema de información, aplicación de sistemas, etc, sin imporatr el uso de sisteams operativos, arquitecturas , lenguajes de programación. Todo ello a través de protocolos y un mismo lenguaje donde los sistemas exista un lenguaje común.
De esta forma conocer la definición de Web Service como aplicación tecnologica y su relación con UDDI hacia donde gira el futuro aplicativo,de desarrollo y cuales son las caracteruisticas ha mostrar de cada uno de los componentes , como utilizarlos y cual es su participación entre ellos.

CAPITULO II
Utilización de software comercial (websphere).

Presentar aspectos relevantes que provee un servidor de aplicación de tipo comercial., con el propósito de obtener un conocimiento previo y tener una medida en la cual pueda comparar características con el software libre.

CAPITULO III
Implementacion de un webservice utilizando jboss.

Para probar un Web Service se ha de tomar una aplicación, ésta consiste en un banco en línea con ciertas funcionalidades básicas que permite acceder a diversos clientes que poseen un cuenta y desean hacer transacciones como: depósitos, retiros, consultar saldos, traspaso entre cuentas, así como proveer un estado de su cuenta., todo basado en un sistemas autentificación y un soporte de seguridad que posee la misma aplicación y también el servidor de aplicación a utilizar.

La aplicación basada en arquitectura J2EE, sistema operativo , servidor de aplicación, herramientas de desarrollo de WebServices y de aplicaciónes, todas ellas en sofware libre. Tendrán la función de ser la base de implementación de un WebServices tomado de los módulos o partes de la aplicación.

El servidor de aplicación JBoss surge como una opción atractiva para estudiar el comportamiento de la aplicación y la posterior implementación de un Web Service con alguno de sus módulos, así también, incluir herramientas de software libre por parte de un grupo importante de desarrollo de software libre como lo es Apache y uno de sus proyectos Ant , el cual proporciona la implementación, instalación y despliegue de la aplicación bancaria, y también, por parte del proyecto Axis, que proporciona la base del Web Service que es SOAP además de una serie de clases precompiladas que dan la funcionalidad y puesta en marcha de el Web Service, incluso se observar que las versiones recientes de JBoss poseen ciertas características orientadas a esta tecnología.

Se ha de utilizar sofware adicional para crear programas clientes para el acceso al WebServices capaz de comprobar el comportamiento que presenta. Visual Studio .Net es un ejemplo de ello, de igual forma existe la capacidad de crear un cliente java pero el objetivo es utilizar clientes capaces de de tener una arquitectura y lenguaje de programación distinta capaces de comunicarse con una aplicación tipo java,asi también es posinble crear un cliente en php, perl, python etc..

Es indispensable aclarar que el WebService ha de estar en un sistema operativo como es Linux, Debido a que las pruebas se hace de forma local se ha de utilizar a su vez el sistema operativo Windows XP, como

consecuencia de que la aplicación requiere interactuar con herramientas de Microsoft para el caso del cliente Visual Basic :NET.

CAPITULO IV

Estudio de Factibilidad del uso de los Web Services

El capítulo permite integrar una visión mas objetiva de esta tecnología, donde se analiza la importancia de un Web Service a nivel negocio, cuales son lo beneficios a corto y mediano plazo, principalmente si una empresa que desee implementar su sistemas o varios de estos , especificar el propósito y sus reales necesidades que satisfaga la misma de forma efectiva cuidando los costos que implica su desarrollo e implementación.

El ambiente competitivo que existe hoy en día se ha generado en gran parte al uso constante de Tecnologías de Información (TI) basadas en Internet. Esto ha llevado a las empresas a desarrollar nuevas estrategias de negocio, para no solamente adaptarse a este ambiente, sino para lograr una ventaja competitiva en este ambiente.

Para lograr esta adaptación, las empresas se han visto forzadas a realizar análisis exhaustivos para poder combinar las (TI) con los procesos de negocio. Estos análisis y modificaciones han generado innumerables tendencias tales como la reingeniería entre otros. Por otra parte, la combinación de las TI con los procesos de negocio, han originado nuevos conceptos electrónicos (CE) que se manejan en el ambiente electrónico, los cuales se ven inmersos en los negocios electrónicos (e-business, e-commerce, business-to-business, etc.), empresas no lucrativas y organizaciones no gubernamentales.

Surge entonces, la necesidad de adaptar las TI y los CE de acuerdo a las necesidades y giros de cada empresa. La forma en que las empresas han logrado involucrar las TI con dichas necesidades es a través de nuevos servicios basados en Internet (SBI) que utilizan tanto dentro y fuera de su negocio, para lograr una diferenciación en el mercado.

La implementación de los (SBI) ha impacto fuertemente a la empresas debido a que en ocasiones, las empresas no se encuentran preparadas para un cambio en sus procesos de negocio. Por lo tanto, es importante realizar previamente una planeación estratégica de los (SBI) que apoyarán a una empresa a alcanzar la ventaja competitiva deseada.

Dentro de estos SBI encontramos la aplicación de los Web Services como una capa más que puede considerarse como alternativa tecnológica para proveer beneficios, innovación, productividad a los procesos de los negocios electrónicos y TI.

Se entenderá las ventajas y desventajas en cuanto al software a utilizar con base a los capítulos anteriores hacer un análisis que proporciona un desarrollo en software libre y el soporte que provee un software comercial por parte de WebSphere Studio de IBM entre otras compañías importantes que están brindando también apoyo a difundir y apoyar el uso de Web Service.

Analizar la alternativa de un Web Service con el uso de servidores de aplicación libre, cuales son los aspectos a considerar en cuanto a uso de versiones, que herramientas adicionales hacen posible el desarrollo e implementación de un Web Services . Definir el soporte en cuanto a seguridad, escalabilidad, facilidad de uso.

Por último discernir, criticar, profundizar, con la finalidad de mantener una aplicación publicada, invocada y localizada por cualquier tipo de sistemas sin necesidad de buscar otros esquemas o bien posiblemente migrar nuestros sistemas o cambiar nuestras plataformas para conseguir el proposito que busca un Web Service.

I. CONCEPTOS, PRINCIPIOS Y TEMAS RELACIONADOS CON LOS WEBSERVICES

1.1 ANTECEDENTE DE LOS SISTEMAS DISTRIBUIDOS

Históricamente el problema que trata de resolver el cómputo distribuido es el de cómo distribuir la computación entre varios sistemas que trabajan en conjunto para resolver un problema. El concepto abstracto más utilizado en el campo de cómputo distribuido es el RPC (Remote Procedure Calls - llamados de procedimiento remotos). Los RPC permiten a una función remota ser llamada como si se tratara de una local. Los sistemas distribuidos orientados a objetos requieren RPC basados en objetos (ORPC). La historia de la computación distribuida y la de objetos distribuidos es un poco complicada, pero se presenta una cronología de algunos eventos:

- **1987**
  - La compañía Sun Microsystems desarrolla el RPC de cómputo de red abierta (ONC - Open Network Computing) como el mecanismo básico de comunicación para su sistema de archivos de red (NFS - Network File System).
  - La compañía Apollo Computer desarrolla el RPC de su sistema de cómputo en red (NCS - Network Computer System) para su sistema operativo de Dominio.
- **1989**
  - La Fundación de Software Abierto (OSF - Open Software Foundation, ahora conocido como The Open Group) lanzó una convocatoria para un sistema RPC. Fué elegida la propuesta NCS de compañía HP/DEC como el mecanismo RPC para su Ambiente de Cómputo Distribuido (DCE - Distributed Computing Environment).
  - Se conforma el Grupo de Administración de Objetos (OMG - Object Management Group) para coordinar las especificaciones para cómputo distribuido independiente de plataforma y lenguaje. El OMG inició el desarrollo de las especificaciones de una plataforma de objetos distribuidos (CORBA - Common Object Request Broker Architecture).
- **1990**
  - Microsoft basa sus iniciativas de RPC en una versión modificada de DCE/RPC.
- **1991**
  - La OSF libera el DCE 1.0.
  - Se libera CORBA 1.0, con solamente correspondencia del lenguaje C. Se populariza el ORB (Object Request Broker)
- **1996**
  - Microsoft presenta el Modelo de Objeto Componente Distribuido (DCOM - Distributed Component Object Model) cercano a la familia de componentes de Microsoft como el OLE (Object Linking and Embedding), COM no distribuido (OLE2) y ActiveX. La parte central de las habilidades de DCOM estan basadas en las tecnologías RPC de Microsoft. DCOM es un protocolo ORPC.
  - CORBA 2.0 es presentado con muchas mejoras en el modelo de cómputo distribuido, en servicios de alto nivel que pueden utilizar los objetos distribuidos. El Protocolo de Internet

Inter-ORG (IIOP - Internet Inter-ORB) presentado permite la interoperación de múltiples ORB's.

- **1997**
  - La compañía Sun Microsystems presenta la herramienta de desarrollo de Java (JDK 1.1) que incluye el Método de Invocación Remota (RMI - Remote Invocation Method). El RMI presenta un modelo de cómputo distribuido utilizando objetos de Java. El RMI es similar al CORBA y DCOM, pero sólo trabaja con objetos Java. El RMI es un protocolo ORPC llamado Protocolo de Método Remoto de Java (JRMP - Java Remote Method Protocol).
  - Microsoft anuncia su primera versión de COM+, el sucesor de DCOM. Las características de COM+ se acercan más al modelo CORBA de cómputo distribuido.
- **1998**
  - Sun Microsystems presenta J2EE (Java 2 Platform Enterprise Edition). La plataforma de Java 2 integra el RMI con IIOP, facilitando la interoperación entre sistemas Java y CORBA.
  - Aparece el Protocolo de Acceso de Objeto Simple (SOAP - Simple Object Access Protocol). Se inicia la era de los servicios Web.

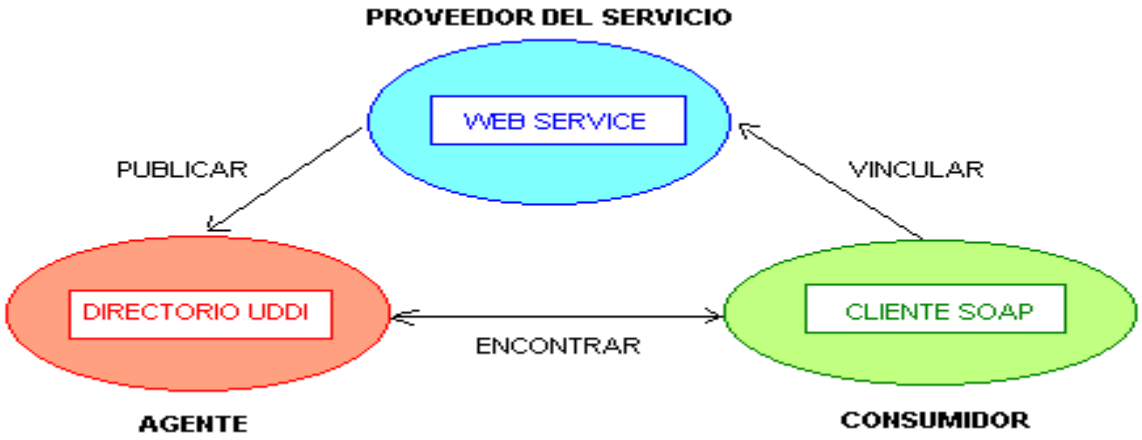
1.2 INTRODUCCIÓN A LOS WEB SERVICES.

Las aplicaciones web actuales ya no son suficientes. El modelo actual de negocio electrónico no facilita la integración de las aplicaciones de Internet con el resto de software de las empresas. Si las compañías quieren extraer el máximo beneficio de Internet, los sitios web deben evolucionar. Este es el contexto en el que surgen los web services.

Los WebServices son componentes software que permiten a los usuarios usar aplicaciones de negocio que comparten datos con otros programas modulares, vía Internet. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos web estándar, como XML, SOAP, UDDI o WSDL. El objetivo final es la creación de un directorio en línea de Web - Services, que pueda ser localizado de un modo sencillo y que tenga una alta fiabilidad.

Los **WebServices** constituyen el siguiente paso en la evolución de la tecnología orientada a objetos, y representan una revolución al alejarse de las arquitecturas tradicionales tipo *cliente-servidor* a nuevas arquitecturas distribuidas tipo *igual-a-igual* (*peer-to-peer).* Estos servicios consisten de un conjunto de estándares que permiten a los desarrolladores implementar aplicaciones distribuidas, utilizando herramientas muy distintas para crear aplicaciones que utilizan una combinación de módulos de software que son llamados desde diversos sistemas distribuidos en regiones geográficas distintas.

La arquitectura de los servicios Web es una meta-arquitectura que permite que ciertos servicios de red sean dinámicamente descritos, publicados, descubiertos e invocados en un ambiente de cómputo distribuido.



Componentes WebServices

Las **componentes** de los servicios Web son:

- **Servicio.** La aplicación es ofrecida para ser utilizada por solicitantes que llenan los requisitos especificados por el proveedor de servicios. La implementación se realiza sobre una plataforma accesible en la red. El servicio se describe a través de un leguaje de descripción de servicio. Tanto la descripción como las políticas de uso han sido publicadas de antemano en un registro.
- **Proveedor de Servicio.** Desde el punto de vista comercial, es quien presta el servicio. Desde el punto de vista de arquitectura, es la plataforma que provee el servicio.
- **Registro de Servicios**. Es un depósito de descripciones de servicios que puede ser consultado, donde los proveedores de servicios publican sus servicios y los solicitantes encuentran los servicios y detalles para utilizar dichos servicios.
- **Solicitante de servicios.** Desde el punto de vista comercial, la empresa que requiere cierto servicio. Desde el punto de vista de la arquitectura, la aplicación o cliente que busca e invoca un servicio.

Operaciones de Servicios Web:

- **Publicar / Cancelar.** Los proveedores de servicios publican (publicitan) la disponibilidad de su servicio comercial (*e*-business) a uno o más registros de servicios, o cancelan la publicación de su servicio.
- **Búsqueda**. Los solicitantes de servicios interactúan con uno o más registros de servicios para descubrir un conjunto de servicios comerciales con los que pueden interactuar para encontrar una solución.
- **Ligar, Unir (Bind)**. Los solicitantes de servicios negocian con los proveedores de servicios para acceder e invocar servicios comerciales (*e*-business)

La funcionalidad de los protocolos empleados es la siguiente:

- **XML(eXtensible Markup Language):** Un servicio web es una aplicación web creada en XML.

- **WSDL (Web Services Definition Service):** Este protocolo se encarga de describir el web service cuando es publicado. Es el lenguaje XML que los proveedores emplean para describir sus web services.

- **SOAP (Simple Object Access Protocol):** Permite que programas que corren en diferentes sistemas operativos se comuniquen. La comunicación entre las diferentes entidades se realiza mediante mensajes que son ruteados mediante el protocolo SOAP.

- **UDDI (Universal Description Discovery and Integration):** Este protocolo permite la publicación y localización de los servicios. Los directorios UDDI actúan como una guía telefónica de web services.

Aunque la idea de la programación modular no es nueva, el éxito de esta tecnología reside en que se basa en estándares conocidos en los que ya se tiene una gran confianza, como el XML. Además, el uso de los web services aporta ventajas significativas a las empresas. El principal objetivo que se logra, es la interoperabilidad y la integración. Mediante los web services, las empresas pueden compartir servicios software con sus clientes y sus socios de negocio. Esto ayudará a las compañías a escalar sus negocios, reduciendo el coste en desarrollo y mantenimiento de software, y sacando los productos al mercado con mayor rapidez. La integración de aplicaciones hará posible obtener la información demandada en tiempo real, acelerando el proceso de toma de decisiones. La evolución de Internet hacia los web services, mejorará los resultados globales de las empresas, reduciendo sus gastos y guiándolas hacia una mejora progresiva de la calidad. La adopción de la tecnología de servicios web por la industria es el primer paso hacia una economía global.

Un servicio Web puede agregar otros servicios Web para proveer un conjunto de mayores características. Por ejemplo, un servicio Web pudiera proveer características de alto nivel en viajes, al orquestar servicios Web de bajo nivel de renta de autos, boletos de avión, y hoteles. La aplicaciones futuras se realizarán con servicios Web que serán seleccionados dinámicamente en tiempo real basados en costo, calidad, y disponibilidad.

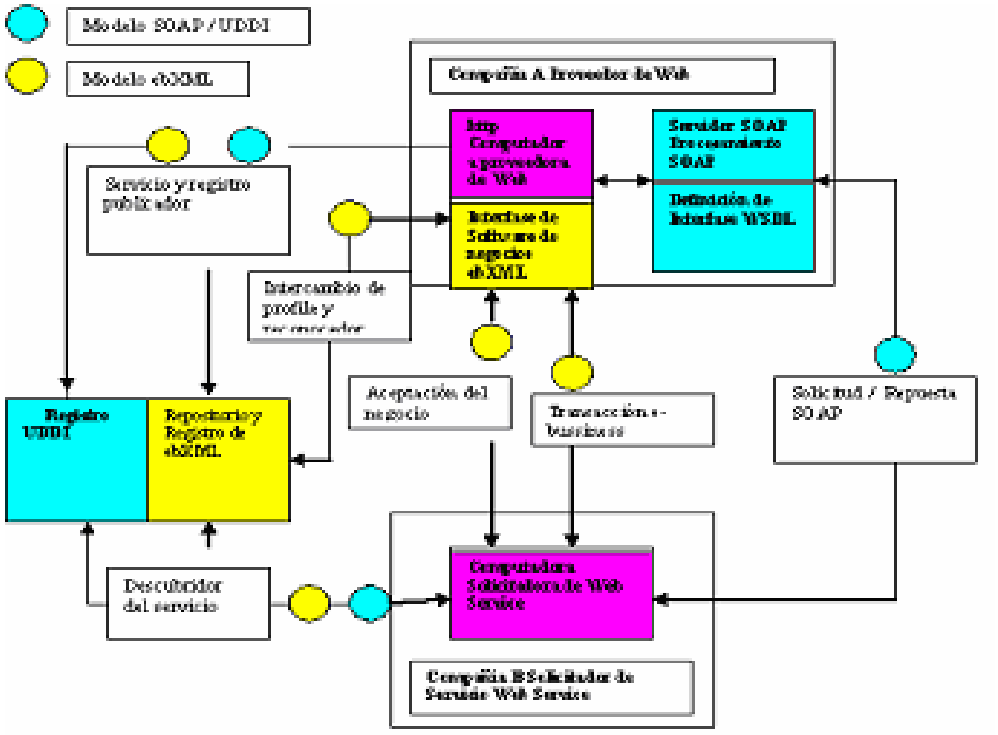
Entre las razones por las cuales los servicios Web jugarán un rol principal en la siguiente generación de sistemas distribuidos, están:

- ***Interoperabilidad.*** Cualquier servicio Web puede interactuar con cualquier otro servicio Web. El protocolo estándar *SOAP* permite que cualquier servicio pueda ser ofrecido o utilizado independientemente del lenguaje o ambiente en que se haya desarrollado.
- ***Omnipresencia.*** Los servicios Web se comunican utilizando *HTTP* y *XML.* Cualquier dispositivo que trabaje con éstas tecnologías puede tanto ser huésped y acceder a los servicios Web. Por ejemplo, pronto serán utilizados en teléfonos, automóviles o aún en máquinas vendedoras de refrescos. Por ejemplo, una máquina de venta de refrescos puede comunicarse vía inalámbrica con el servicio Web de un proveedor local y ordenar un pedido de suministro.
- ***Barrera mínima de participación.*** Los conceptos detrás de los servicios de Web son fáciles de comprender y se ofrecen Herramientas de Desarrollo (*ToolKits*) por IBM, Sun Microsystems y la Organización de Apache, permiten a los desarrolladores crear e implementar rápidamente servicios de Web.

Apoyo de las Industrias. Todas las compañías apoyan el protocolo *SOAP* y la tecnología derivada de los servicios Web. La funcionalidad de los protocolos empleados es la siguiente:

1.2.1 Implementando un WebServices

Se presenta un breve esquema paso a paso de cómo se implementa un servicio Web en el siguiente diagrama:



Esquema Implementación WebServices

1. Un proveedor de servicio crea un servicio Web
2. El proveedor de servicio utiliza WSDL para describir el servicio (a un registro UDDI)
3. El proveedor de servicio registra el servicio (en registro UDDI y/o a un registro/depósito ebXML)
4. Otro servicio o usuario localiza y solicita el servicio registrado al consultar los registros UDDI.
5. El servicio o usuario solicitante escribe una aplicación que liga el servicio registrado utilizando SOAP (en el caso de UDDI).
6. Se intercambian datos y mensajes XML sobre HTTP

1.3 INTRODUCCIÓN SOAP

Este protocolo representa el principal camino de información o comunicación entre los tres principales actores en SOA (service-oriented architecture): servicio proveedor, solicitador del servicio e intermediario de servicio.

SOAP (Simple Object Access Protocol) es un protocolo basado en XML el cual consiste de 3 partes: Una cubierta que describe un esqueleto para describir el contenido de mensajes y procesar instrucciones. Un conjunto de reglas para explicitas instancias de tipo de datos de aplicación definidas y una convención para representar llamadas de procesamiento remoto y respuestas.

SOAP es un protocolo independiente del transporte fundamental (HTTP) y por lo tanto puede ser usado potencialmente en combinación con una variedad de protocolos, mencionaremos como usar el SOAP con

http, http extendido a la estructura y JMS. SOAP es un protocolo independiente del sistema operativo y no esta ligado a utilizar un lenguaje de programación o componente de tecnología por ser un protocolo estándar.

SOAP no abarca la distribución de recolección de basura. SOAP cliente no toma ninguna referencia total del estado a un objeto remoto y sin esto, la activación en un medio java RMI es también posible.

Debido a estas características, la tecnología no importa para la implementación hacia los clientes: El cliente podrá editar mensajes en XML. Similarmente el servicio podrá ser implementado en cualquier lenguaje y se podrán procesar mensajes en XML. También ambos servidor como cliente podrán residir en la plataforma que les convenga.

1.3.1 INTERCAMBIO DE MENSAJES EN SOAP.

SOAP estándar especifica 3 aspectos de intercambio de mensajes basados en XML.

FORMATO DE MENSAJES EN CONJUNTO

- Un mensaje SOAP es una cubierta contenida de cero o mas cabeceras y solo un cuerpo. La envoltura es el elemento superior del documento XML, suministra un contenedor para el control de información, el destinatario de un mensaje y el mensaje propio.

- Las cabeceras contienen información de control así como características de los atributos del servicio, el cuerpo contiene la identificación del mensaje y sus parámetros.

- En la estructura SOAP cabecera y cuerpo son elementos pequeños de la cubierta.

REGLAS CODIFICADAS

- Las reglas codificadas definen un mecanismo de serialización que puede ser usado en el intercambio de instancias de tipos de datos de la aplicación definida.

- SOAP define una programación independiente del lenguaje del esquema de los tipos de datos basado sobre el esquema XML Descriptor (XSD), la cantidad de reglas de código para todos los tipos de datos definidos de acuerdo a este modelo.

REPRESENTACION RPC

- RPC (llamada de procesamiento remoto) Como argumento es un método remoto de invocación donde utilizaremos estructura de datos simples, además, con convenciones como lo es XML literal, esto es posible para transferir datos mas complejos, esta convención, sin embargo, es solo cubierta por implementaciones SOAP y por lo tanto no es parte de SOAP estándar.

- La utilización de esta convención es opcional. Si RPC no es usada el estilo de comunicación es puramente un mensaje-orientado , cuando se trabaja con un mensaje-orientado, también llamado comunicación de documento-orientado casi ningún documento XML podrá ser intercambiado.

- El estándar de cabeceras http contiene la URL del servidor SOAP, como en este caso es /www.messages.com/servlet/rpcrouter. Relativo a esta URL, el Web Services es identificado por urn:NextMessage.

- Después la cabecera llega a una cubierta SOAP que contiene el mensaje a ser transmitido. Aquí el mensaje de invocación es el SOAPRPC representación de una llamada al método GETMENSAJE(ID_USUARIO, CONTRASEÑA) de un Web Services llamado urn:NextMessage residente sobre el servidor SOAP.

- HTTP::///SCHEMAS.XMLSOAP.ORG/ENCODING/ especifica la codificación que es usada para convertir el valor del parámetro para el lenguaje de programación en ambos lados tanto en el lado del cliente como en el lado del servidor a XML y viceversa.

  Ver código de mensaje SOAP embebido en una solicitud http. (Anexo I Capitulo I inciso a).

- Primeras llegadas de la cabecera estándar HTTP.

- Después de que la cabecera llega a la cubierta SOAP que contiene el mensaje a ser transmitido. En este mensaje, el servicio regresa el mensaje solicitado

  Ver código de un mensaje SOAP embebido en una respuesta http. (Anexo I Capitulo I inciso b).

Los mensajes SOAP fundamentalmente son un medio de transmisión desde un emisor a un receptor, pero es solo ilustrado, mensajes SOAP son frecuentemente combinados a patrones de implementación como solicitud / respuesta.

NOMBRES DE ESPACIO (NAMESPACE)

SOAP define los nombres de espacio de XML mostrados en la siguiente tabla:

Nombres de espacio SOAP

Prefijo	URI Espacio de nombres	Explicación
SOAP-ENV	**http://schemas.xmlsoap.org/soap/envelope**	**Envoltura SOAP**
SOAP-ENC	**http://schemas.xmlsoap.org/soap/encoding**	**Serializacion SOAP**

URN

Por sus siglas nombre de recurso unificado (unified resource name) excepcionalmente identifica el servicio a clientes, esto debería ser unificado entre todos los servicios desplegados en un simple servidor SOAP, el cual es identificado por una dirección de red confirmada. Una URN es codificada como un identificador de recursos universal (URI). Aquí se usa el formato urn:UniqueServiceID. Urn:NextMessage es la URL de nuestro mensaje intercambiado en el Web Service.

CUBIERTA SOAP

En general un mensaje SOAP es un conjunto (posiblemente vació) de cabeceras mas un cuerpo: La cubierta SOAP también define el nombre de espacio para estructurar mensajes. El mensaje completo SOAP (cabecera y cuerpo) es agregado en esta cubierta.

Nota que el cuerpo del mensaje usa un servicio de espacio de nombres especifico, urn:NextMessage. Por consiguiente, la aplicación puede ser usada por su propio vocabulario de dominio-especificado cuando se crean mensajes en el cuerpo.

CABECERAS

Las cabeceras son un mecanismo genérico y flexible para entender un mensaje SOAP en una forma descentralizada y modular sin previo acuerdo o arreglo entre las partes involucradas. Ello permite controlar la información a pasar a la recepción del servidor SOAP proveer extensibilidad para mensajes estructurados igualmente. Las cabeceras son elementos opcionales en la cubierta.

Un SOAP intermediario es una aplicación capaz tanto de recibir y enviar mensajes SOAP sobre su forma al destinatario final. En situaciones reales, no todas las partes del mensaje SOAP podrían ser deseados por la ultima destinación del mensaje SOAP, pero, en cambio podrían ser deseados por uno o mas de los intermediarios sobre la ruta del mensaje. El segundo atributo de encabezado SOAP-ENV: actor, es usado para identificar al receptor de la información del encabezado. El valor del atributo actor del SOAP es la URI del mediador, el cual es también el destinatario final del elemento del encabezado particular (el mediador no envía el encabezado).

Las cabeceras también pueden llevar datos de autentificación, firmas digitales, información encriptada y ambientación transaccional.

Las cabeceras también pueden llevar clientes o controles de proyectos específicos y extensiones del protocolo, la definición de cabeceras no es apropiada para cuerpos estándar.

CUERPO

El elemento del cuerpo SOAP suministra un mecanismo para intercambiar información prevista para el ultimo almacén del mensaje. El elemento del cuerpo es codificado como un pequeño elemento inmediato del elemento de la cubierta SOAP. Si un elemento del encabezado esta presente, entonces el elemento del cuerpo debería seguir inmediatamente del elemento del encabezado. Otro caso debería ser el primer elemento inmediato pequeño del elemento de la cubierta.

Todos los pequeños elementos inmediatos del elemento del cuerpo son llamados las entradas del cuerpo y cada entrada del cuerpo es codificada como un elemento independiente dentro del elemento del cuerpo SOAP. En el caso mas simple, el cuerpo de un mensaje básico del SOAP comprende:

- Nombre del mensaje
- Una referencia a una instancia de servicio. En apache SOAP, una instancia de servicio es identificada por una URN: Esta referencia se codifica como el atributo del nombre de espacio.
- Uno o mas parámetros que llevan los valores y opcionalmente referencias del tipo

Típicamente las aplicaciones del elemento del cuerpo incluyen la invocación de llamadas RPC con parámetros apropiados, regresando resultados y reportes de error. Los elementos de avería son utilizados en situaciones de error que se comunican. Los mensajes pueden contener casi cualquier constructor de XML excepto documentos de tipo de definición (DTD) y el procesamiento de instrucciones.

1.3.2 ASUNTOS AVANZADOS DE LA COMUNICACIÓN SOAP.

ESTILOS DE COMUNICACIÓN SOAP

SOAP soporta dos diferentes estilos de comunicación.

DOCUMENT.- Conocido también como estilo de mensaje-orientado. Este proporciona una capa baja de abstracción y requiere mas trabajo de programación. Un parámetro es cualquier documento XML, la respuesta puede ser cualquier cosa (o nada). Este estilo es muy flexible pero no usado con frecuencia en la comunicación.

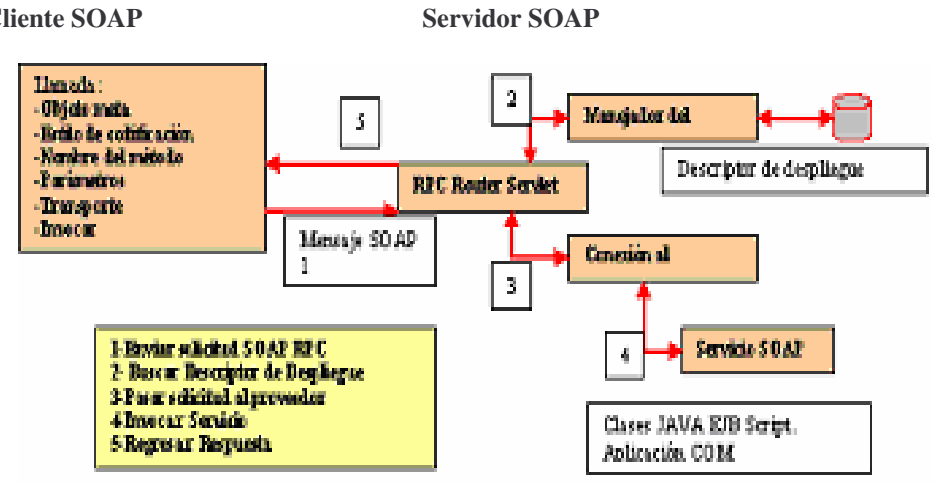
RPC.- La llamada de procesamiento remoto es una invocación sincrona de la operación que devuelva un resultado conceptualmente o otros RPC's. Este estilo es explotado por muchas herramientas del Web Services.

SOAP permite aplicaciones encapsuladas e intercambia llamadas al RPC usando la extensibilidad y flexibilidad de **XML**. Para hacer una llamada al método la información siguiente es necesaria:

- La **URI** del objeto en cuestión.
- Un nombre del método.
- Una firma opcional del método.
- Los parámetros del método.
- Datos opcionales de encabezado.

Usar el SOAP para RPC no limita al uso de solo el protocolo http, debido a que se efectúa de la siguiente forma una llamada a RPC tras una petición del http y a los mapeos de una respuesta del RPC a una respuesta del http.

Se muestra el diagrama de interacción entre el cliente y el servidor así como el lado del servidor procesando un servicio de invocación en comunicación en RPC.



Interacción SOAP entre cliente y servidor.

Web Services usando invocación RPC envuelve los siguientes pasos:

1.- Un cliente SOAP genera el documento de la petición del RPC del SOAP y lo envía a un ruteador del RPC.
2.- El ruteador entra en contacto con el manejador de servicio para obtener un descriptor del despliegue.
3.- De acuerdo con el enrutamiento de la información del descriptor de despliegue el ruteador transmite la solicitud a un proveedor de servicio.
4.- El proveedor de servicio invoca el servicio solicitado y devuelve un resultado al ruteador.
5.- El ruteador envía el mensaje resultado del servicio al cliente.

1.3.3 CODIFICACION

Define como los valores de los datos definidos en la aplicación se puede traducir a y desde un formato de protocolo.

Al implementar un Web Services se tiene que elegir una de las dos herramientas y programar en un lenguaje o script que apoye el modelo Web Services, por ejemplo java. Por otra parte, el formato del protocolo para el Web Services es XML, el cual es independiente del lenguaje de programación. Así, la codificación del SOAP dice al ambiente en tiempo de ejecución SOAP como traducir las estructuras de datos construidas en un lenguaje de programación específico en el SOAP XML y viceversa.

Se define la siguiente codificación:

- **CODIFICACION SOAP.-** La codificación habilitada SOAP formada /deformada de valores de tipo de datos del modelo de SOAP. Esta codificación se define en el estándar del SOAP 1.1.

- **LITERAL XML.-** La codificación literal XML permite la conversión directa de los elementos existentes del árbol XMLDOM dentro del contenido del mensaje SOAP y viceversa. Este tipo de codificación no es definida por el estándar SOAP, pero esta en la implementación del apache SOAP.

- **XMI.-** El intercambio de meta-datos de XML (XMI) es definido en la implementación del apache SOAP. No se utiliza esta codificación en este documento.

La codificación que se utilizara en tiempo de ejecución del SOAP se podrá especificar en tiempo de despliegue o de ejecución si se especifica en tiempo de despliegue. Esta aparecerá en la especificación WSDL del Web Services. La herramienta puede ser analizada, especificada y configurada en tiempo de ejecución del SOAP para utilizar la codificación deseada.

En tiempo de ejecución el cliente SOAP API permite la especificación de la codificación para el mensaje completo y para cada uno de sus parámetros. En el lado del servidor, el estilo de codificación es especificada en el descriptor de despliegue del Web Services.

En el documento XML que proporcione la información en el tiempo de ejecución del SOAP, sobre los servicios que se deben poner a disposición de los clientes. Esto contiene la información como la URN para el servicio, método y los detalles de la clase (en caso de que el servicio se implemente en java) o el nombre de un script. La configuración en el lado del cliente y en el lado del servidor tiene que corresponder.

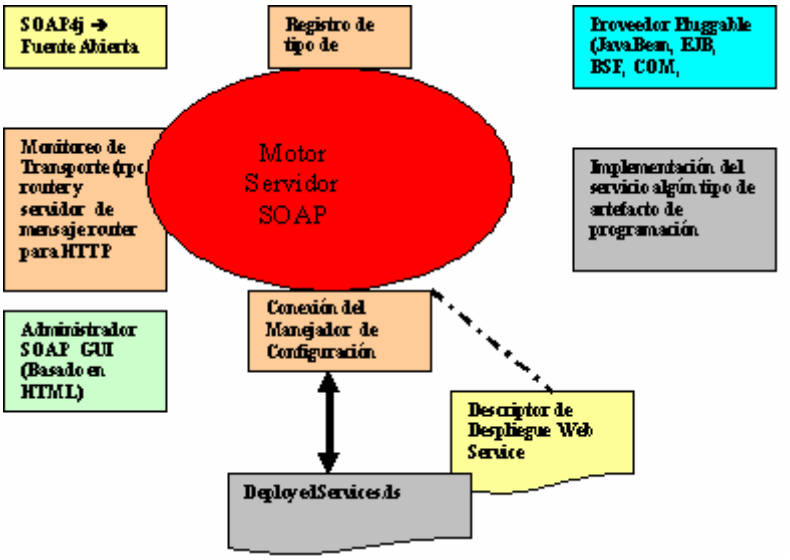
Ver la arquitectura general de una implementación SOAP. (Anexo I Capitulo I inciso b).

El cliente puede invocar SOAP mensajes a través de la capa RPC (Estilo RPC) o directamente contra la capa del mensaje (Estilo del documento) varios protocolos de transporte tales como http, SMTP y otros conectan al solicitante y al proveedor.

En el lado del proveedor, el RPC y los mensajes servlet ruteados reciben las peticiones entrantes del mensaje.

Los proveedores rutean a la implementación del servicio de java. El servidor es configurado a través de archivos del despliegue.

En el solicitante y en el lado del servidor, ahí están los registros mapeados que proporcionan acceso a clases de serializar / deserializar que implementan un esquema de codificación.



Componentes de implementación del servidor apache SOAP. .

1.3.4 RESUMEN SOAP

SOAP representa el mecanismo de información intercambiada entre el principal árbol actor en el modelo Web Services: un Web Services proveedor, un Web Service solicitador y un Web Service intermediario esto es basado en documentos XML y es diseñado para ser simple y extensible. Esto también es independiente de la implementación Web Services y por consiguiente permite la Inter-Operatibilidad entre la implementación Web Services en diferentes plataformas SOAP definido por W3C**.**

SOAP permite la publicación e invocación de Web Services y representa el conducto básico de la infraestructura de un Web Services, sin embargo, esto no es suficiente para la implementación exitosa de un Web Services.

- Un cliente tiene que obtener la información sobre la interfase de un servicio, el servidor URL, el URN, los métodos, los tipos, y tipo de mapeo. Esto es suministrado por WSDL.

- Un cliente tendrá que saber sobre la existencia de un servicio y sus características, lo cual es la función de UDDI.

- Un documento de lenguaje de inspección Web Services (WSIL) provee información de localización para invocar Web Services.

1.4 INTRODUCCIÓN A WSDL

WSDL permite a un servicio proveedor especificar las siguientes características de un Web Services:

-Nombre del Web Services y dirección de la información.
-Protocolo y estilo de codificación para ser usada cuando se acceda las operaciones publicas de un servicio.
-Tipo de información: operaciones, parámetros, y tipos de datos comprendiendo la interfase del Web Service, mas un nombre para esta interfase.

Una especificación WSDL usa una sintaxis XML; por consiguiente, ahí se trata de un esquema. Un documento valido para WSDL consiste de uno o más archivos. Si ahí es mas que un archivo, el uso de importantes elementos es requerido. Este elemento importante crea la necesidad de referirse a los diferentes archivos del documento WSDL.

1.4.1 DOCUMENTO WSDL

El documento WSDL contiene los siguientes elementos de nivel-superior:

Types (Tipos).- Un contenedor para la definición de tipos de datos usando algunos tipos de sistemas tales como el esquema XML.

Message (Mensaje).- Un abstracto, tipo de definición de los datos que son comunicados. Un mensaje puede tener uno o más tipos de componentes.

Port type (Tipo de Puerto).- Un conjunto abstracto de una o mas operaciones soportados por uno o mas puertos. Cada una de las operaciones define una entrada y una salida de mensaje así como es un mensaje opcional por default.

Operation (Operacion).- Una descripción abstracta de una acción soportada por el servicio.

Binding.- Un protocolo concreto y especificación de formato de datos para un particular tipo de puertos. La información binding contiene el nombre del protocolo, el estilo de invocación, un servicio ID, y la codificación para cada operación.

Port (Puerto).- Un simple punto final, que es definido como una agregación de un binding y una dirección de red.

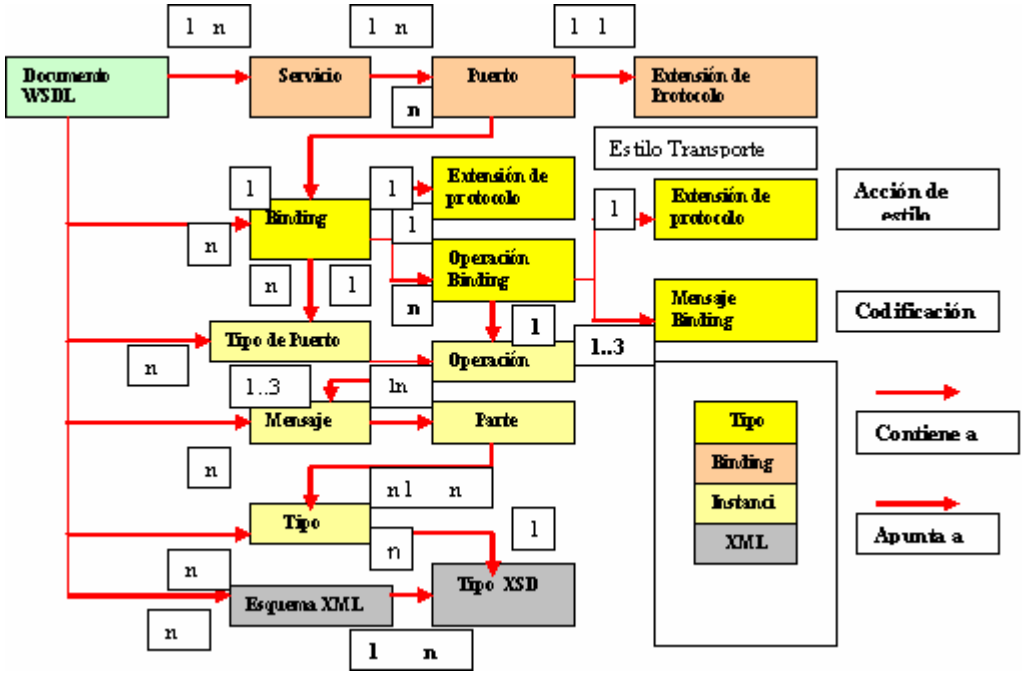
Service (Servicio).- Una colección de puertos relacionados.

El WSDL no introduce un nuevo tipo de lenguaje de definiciones. WSDL identifica la necesidad para el abundante tipo de sistemas, para describir formatos de mensaje, y soporte para la especificación del esquema XML (XSD).

Aquí es un SOAP binding WSDL, que es capas de describir SOAP sobre http, una directa interfase http y un MIME binding. EL lenguaje es extendido y permite la definición de otros mecanismos binding, tales como JAVA y SOAP sobre Servicio de Mensajes Java (JMS)

1.4.2 ANATOMIA DE LOS DOCUMENTOS WSDL.

Abajo se muestra los elementos que comprenden un documento WSDL así como las diversas relaciones entre ellos.



Elementos WSDL y sus relaciones.

El diagrama debe ser leído de la siguiente forma:

- Un documento WSDL contiene cero o más servicios. Un servicio contiene cero o más definiciones de puertos (servicio terminal), y una definición de puerto contiene un específico protocolo de extensión.

- El mismo documento WSDL contiene cero o más bindings. Un binding es referenciado por cero o más puertos. El binding contiene una extensión de protocolo, donde el estilo y el transporte son definidos, y cero o mas bindings de operación. Cada una de estas operaciones bindings es compuesta de una extensión de protocolo, donde la acción y estilo son definidos, y de uno a tres mensajes binding, donde la codificación es definida.

- El mismo documento WSDL contiene cero o más tipos de puertos. Un tipo de puerto es referenciado por cero o más bindings. Este tipo de puerto contiene cero o más operaciones, las cuales son referenciadas por cero o más operaciones bindings.

- El mismo documento contiene cero o más mensajes. De uno a tres mensajes son solicitados para definir una operación. El mensaje es compuesto de cero o más partes.

- El mismo documento WSDL contiene cero o más tipos. Un tipo puede ser referenciado por cero o más partes.

- El mismo documento WSDL apunta a cero o más esquemas WSDL. Un esquema XML contiene cero o más tipos XSD que definen los diferentes tipos de datos. La relación contenida mostrada en el diagrama directamente mapea al esquema XML para WSDL**.**

Ver Ejemplo de un documento WSDL (Anexo I I Capitulo II Pag 60 inciso (a) y (b))

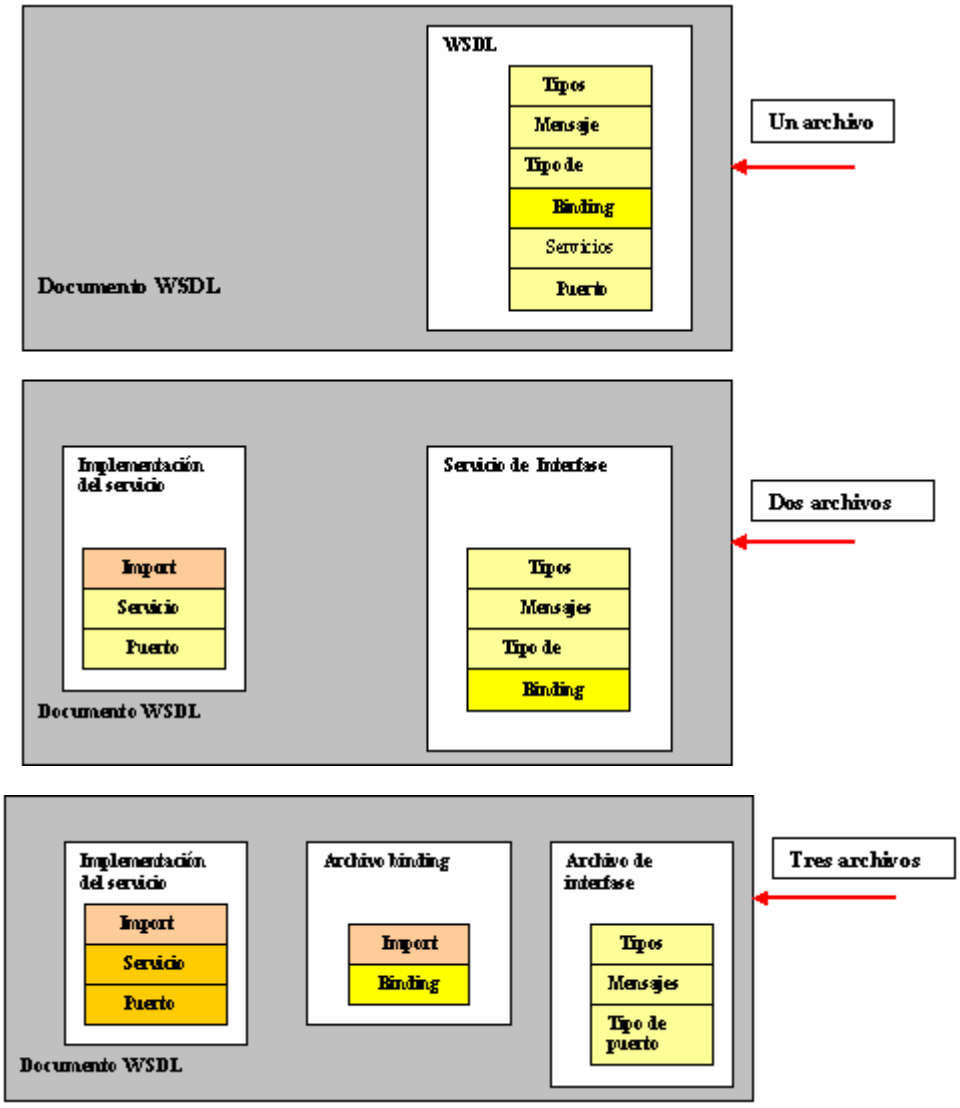
1.4.3 ARCHIVOS FISICOS

Un documento WSDL puede ser creado en uno o más archivos físicos. Si estos son más de uno, se necesitara la conexión de estos archivos usando un elemento. Esta separación de archivos puede ser conveniente para asegurar la abstracción de conceptos y permitir el mejor mantenimiento.

En la Figura mostramos el mismo Web Services descrito en uno, dos o tres archivos.

- Un archivo, típicamente usado en axis.
- Dos archivos, típicamente usados en Application Developer Versión 4
- Tres archivos, típicamente usados en Application Developer Versión 5

Estos tres ejemplos presentados para el mismo Web Service, es importante no confundirse por el numero de archivos usados para definir el documento WSDL. Aquí es solo una especificación WSDL que define los elementos de un documento WSDL; como varios archivos son usados para cargar el documento para hacer la implementación.



Estructura de un documento WSDL como uno, dos o tres archivos. .

WSDL usa los Espacios de nombres XML listados en la tabla (Ver Anexo II Capitulo II Pag. 50)

1.4.4 DEFINICIÓN WSDL

La definición WSDL contiene tipos, mensajes, operaciones, tipos de puertos, bindings, puertos y servicios.

También WSDL provee un elemento opcional llamado wsdl:document como un contenedor para documentación legible por desarrolladores.

TIPOS

Los tipos de elementos incluidos en la definición de tipo de datos usados por el intercambio de mensajes. WSDL usa definición de esquemas XML (XSD's) así su canónico e integrado tipo de sistema.

El tipo de sistema XSD puede ser usado para definir los tipos en un mensaje independiente de si o no el resultado del formato de la liga es actualmente XML.

Aquí es un elemento extensible (área segura para elementos adicionales XML) eso es que puede ser usado para proveer un contenedor de elementos XML para definir adicionalmente tipos de informaciones en caso de que el tipo de sistema XSD no provee suficiente capacidad de modelado.

Ver Ejemplo donde se estructura un elemento con XML (Anexo II Capitulo II Pag. 51 (d))

1.4.5 DEFINICIÓN DEL SERVICIO

Una definición de servicio solamente es un pequeño conjunto de puertos en conjunto bajo un nombre, así como la siguiente definición de seudo XSD del elemento de servicio mostrado. Esta notación seudo XSD es introducida por la especificación WSDL.

```
<wsdl:definitions .... >
   <wsdl:service name="nmtoken"> (0 or more)
      <wsdl:port .... /> (0 or more)
   </wsdl:service>
</wsdl:definitions>
```

Múltiples definiciones de servicios pueden aparecer en un simple documento WSDL.

1.4.6 RESUMEN WSDL

Esta introducción muestra el poder de WSDL, WSDL provee una parte abstracta que es un lenguaje y protocolo independiente, así como un binding para el protocolo usado en tiempo de ejecución en la arquitectura de servicio-orientado (SOA).

También en este capitulo se muestra que hasta una definición de Web Service tiene que cubrir muchos aspectos flexibles y productivos de interrelaciones mas que archivos de especificaciones complejas.

1.5 INTRODUCCION A UDDI

1.5.1 VISIÓN GENERAL UDDI

Especificación que define un camino para almacenar y recuperar información sobre un negocio y sus interfaces técnicas, en este caso Web Services. Una implementación de esta especificación es el registro de negocio UDDI (Universal, Description, Discovery and Integration) o UBR. Este es un grupo basado en base-Web en nodos UDDI que juntos forman un registro UDDI. Estos nodos corren por distintos sitios y pueden ser usados por cualquier persona que quiera dar de alta información disponible acerca de un negocio o entidad, así como cualquier persona que desee encontrar esa información.

El registro UDDI hace posible descubrir que interfaces de programación técnica se proporciona para interactuar con un negocio para propósitos tales como comercio-electrónico, recuperación de información, etc.

UDDI ayuda a ampliar y simplificar la interacción de negocio a negocio B2B. Para el fabricante que necesita crear muchas relaciones con diferentes clientes, cada uno con su propio sistema de estándares y de protocolos. UDDI proporciona una descripción altamente flexible de servicios usando virtualmente cualquier interfaz. La especificación permite el eficiente y sencillo descubrimiento de otros servicios y de los servicios que ofrece para la publicación de los negocios en el registro.

UDDI esta basado sobre un estándar existente, como es XML o SOAP. Esto es una capa descubierta técnicamente, esto es definido:

- La estructura para un registro de proveedores de servicio y servicios.
- El API que puede ser usado para acceder registros con esta estructura.
- La organización y el proyecto definiendo esta estructura de registro y esta es una API.

En realidad UDDI es un motor de búsqueda para la aplicación cliente, mejor que la búsqueda de un ser humano (hombre que realiza una búsqueda de negocios). Sin embargo, muchas implementaciones suministran una interfase de navegación (browser) para usarlos por los clientes.

La información central de UDDI es operado por OASIS, es un consorcio global que maneja el desarrollo, convergencia y adopción de estándares por parte de los negocios electrónicos.

1.5.2 WEB SERVICES DINAMICOS VS. ESTATICOS

Frecuentemente se lee de Web Services estáticos, lo que significa que el servicio suministrado y el servicio solicitador conocen uno acerca del otro en tiempo de diseño. Aquí es un documento WSDL que será encontrado en un directorio UDDI o – más frecuentemente – directamente enviar a la aplicación cliente desarrolladores para el suministro para proveer su uso sobre la herramienta de desarrollo. Durante el tiempo de ejecución esto es muy transparente (en la mayoría de los casos codificado rígido) con la URL (punto de acceso) del socio.

Web Services dinámicos describe el hecho que en diseño y en tiempo de desarrollo el cliente no conoce. El servidor explícitamente y la entidad de negocio de donde esta invocara el servicio. El cliente solo conoce una interfase para llamar y encontrar fuera uno o mas proveedores concretamente para esta categoría de servicio buscada a través de incursionar los registros UDDI en tiempo de ejecución.

1.5.3 ESTRUCTURA DEL REGISTRO UDDI

Aquí son distintos tipos de información que tienen que ser cargados o almacenados en algo semejante a un registro, incluyendo información acerca de la compañía -en UDDI llamado una entidad de negocio- que ofrece servicios, así como la descripción de cada uno de los servicios incluyendo interfaces de especificación y el apuntador o indicador a servicios donde esos servicios pueden ser invocados.

Los datos hacer almacenados pueden ser divididos dentro de seis tipos de información que incrementan el modelo de datos del registro:

- **Business entity (Entidad de Negocio).-** La lista de entidades de negocio es similar a las paginas blancas y amarillas. Una entidad de negocio describe una compañía u organización. Estas entidades suministran información tal como el nombre del negocio, contactos, descripciones, identificadores y categorización.

- **Business service (Servicio de Negocios).-** Esta no es una información técnica acerca del servicio que es suministrado. Un servicio de negocio es un contenedor descriptivo usado por un grupo fijo de Web Services relacionados para un proceso de negocio o grupo de servicios. También la categorización es disponible sobre este nivel. Un servicio de negocio mapeado hacia un servicio WSDL.

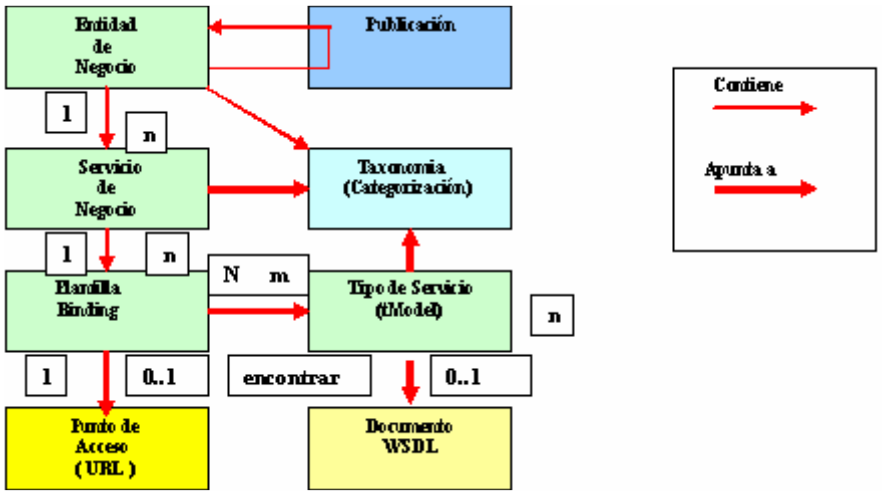
- **Binding template**.- Este servicio contiene información de acceso. Allí se encuentra la descripción técnica de un Web Service relevante para los desarrolladores de aplicaciones quienes desean encontrar e invocar un Web Service. En muchos casos un binding template apunta a una dirección de una implementación (por ejemplo, una URL) y mapea a un puerto WSDL. Esta entidad es algunas veces también llamada un localizador de acceso.

- **tModel.-** Un tModel (modelo técnico) es técnicamente una impresión digital que contiene meta-datos acerca del tipo de especificaciones así como es la categorización de la información. Estos atributos son claves, nombre, descripción opcional, y una URL. El modelo simple contiene algo de texto que caracteriza un servicio. Un tModel es algunas veces referenciado a un tipo de servicio. En este caso, el tModel apunta a un documento de descripción de servicio (por ejemplo, a través de un URL). El tipo de especificación a si mismo, el cual puede ser un documento WSDL o alguna otra especificación formal, esto no es parte del modelo UDDI.

- **Taxonomy (Taxonomia).-** Una taxonomía es un esquema para categorización. Aquí es un conjunto de taxonomias estándar, semejantes al Sistema de Clasificación de la Industria Norte Americana (NAICS) o el estándar Universal de Productos y Servicios en su Clasificación (UNSPSC). En circunstancias específicas puedes editar tu propia taxonomia. Pero esto es típicamente no posible usando el cliente Web o editando API's por que se necesita autorización especial y acción por parte del operador del registro UDDI.

- **Publisher assertions (relación entre negocios).-** También llamado esto relación entre negocios. Aquí son diferentes tipos de relaciones: padre-hijo, par-par, e identidad. Esto hace posible un modelo de negocios complejo, como es una sucursal, socios de negocios externos, o divisiones internas.

La siguiente figura muestra el modelo de datos con las entidades que se citaron anteriormente. Esta también muestra sus relaciones y la liga de documentos WSDL.



Relación de Entidades del modelo de datos UDDI y la liga WSDL.

Entidades UDDI y sus relaciones

La entidad de negocio mostrada contiene una jerarquía, contiene una o más entidades de negocios que ofrecen servicio. Estos servicios en su dirección contienen entidades de plantilla binding. Las instancias de tModel (tipos de servicios) no están contenidos por las entidades de negocio, pero son referidas por las entidades de plantilla binding.

Una plantilla binding apunta a un punto de acceso URL donde el servicio puede ser invocado. Este es un común uso de la plantilla binding, pero no necesario. La plantilla binding podría apuntar a un numero telefónico como si fuera punto de contacto.

Un tipo de servicio de una entidad (tModel) contendrá una referencia a un tipo de especificación de un Web Services el cual típicamente es un documento WSDL. Note que el registro UDDI solamente contiene un URL apuntando al sitio Web del Web Service suministrado donde el documento puede ser encontrado, no el documentos WSDL por si mismo.

Negocios y Servicios y tModels pueden contener cero o muchas referencias a entradas de taxonomias para categorizar su contenido.

Estas referencias son implementadas como URLs. Por lo tanto, cualquier otro formato de especificación puede ser realmente soportado fácilmente también. UDDI no esta en el WSDL. En general se puede no trabajar con archivos WSDL en todo el proceso cuando estas trabajando con registros UDDI.

Aquí es una relación m : n entre la plantilla binding y el tModel. Mantenga en mente que un tModel es exactamente una huella digital técnica que contiene meta-datos abstractos totalmente. Incluso si un servicio apunta a distintos tModels, esto no se tendrá que implementar necesariamente con múltiples interfaces es un nivel técnico.

El modelo de datos UDDI es diseñado para ser flexible. Por ello, allí podrá ser más que una descripción técnica para un servicio (una especificación formal y un tutorial que soporten, por ejemplo). Viceversa, un tModel podrá ser usado para describir distintos servicios de negocios similares.

La posibilidad para el solicitante a un bind dinámico a través de un servicio suministrado a un determinado tModel es un beneficio real de la arquitectura Web Services.

1.5.4 INTERACCIONES CON UDDI

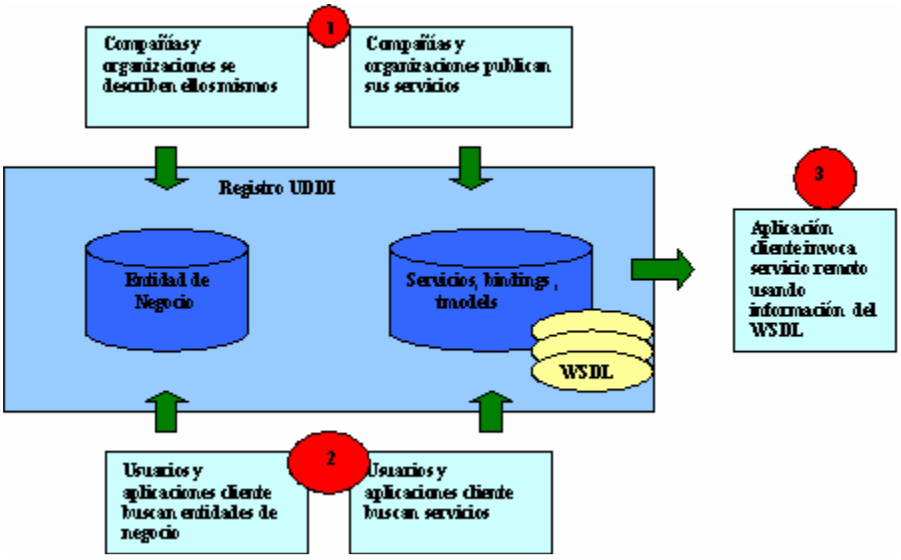
Usando registros UDDI estos contienen en general 3 diferentes tareas, por que por un lado son servicios suministrados para quien desea publicar alguna información, por el otro lado es un servicio solicitante para quien desea encontrar algún servicio y finalmente usarlo. Así que las tareas usadas en un registro UDDI son:

1.- Publicación de información.

2.- Encontrando información.

3.- Usando la información obtenida.

Estos pasos típicos de interacción con un registro UDDI son mostrados en la figura de abajo.



Interacción con registros UDDI.

PUBLICANDO INFORMACION EN LOS REGISTROS UDDI

Abajo se muestran los 6 tipos de información que pueden ser publicados en registros UDDI.

- Información de entidad de negocios
- Información del servicio del negocio
- Plantilla Binding
- tModels
- Información Taxonómica
- Publicar assertions (relaciones de negocios)

ENCONTRANDO INFORMACION EN LOS REGISTROS UDDI

Después de que las entidades de negocios tienen publicados su descripción de entidad y servicios, clientes (solicitador de servicios) podrían tratar de encontrar esta información.

Los clientes tienen un número de posibles caminos para explotar un registro. Una persona hace eso usando clientes HTML del registro o aplicaciones usando API's de UDDI para hacer esto automáticamente.

En cualquier caso la estrategia más probable deberá ser una de las siguientes:

- Encontrar entidades de negocios concretas y explotar sus servicios.
- Encontrar servicios de un cierto tipo sin importar de cual entidad proviene este.

1.5.5 REGISTRO DE NEGOCIOS UDDI EN LA WEB

Actualmente son 4 organizaciones que almacenan o reciben los nodos en el registro de negocio UDDI. Estas 4 organizaciones son replicadas y por lo tanto contienen la misma información. La mayoría de las compañías también hospedan un registro de pruebas además del registro del negocio. Esas instancias de prueba pueden ser usadas por cualquier persona con el propósito de realizar pruebas. Los Web Services encontrados ahí están frecuentemente sujetos a cambios y a la carencia de la disponibilidad. Ello no significa que sea para el uso de la producción.

La siguiente tabla especifica 3 URLs para cada registro. El primero es el URL para el cliente Web (Web client) que puede ser usado por personas para explorar y publicar cualquier información. Usted tiene típicamente acceso ilimitado para la lectura de información y necesita una cierta clase de registro para la información que replica.

Del lado del URL cliente Web (Web client) Usted también encontrara URL's para el acceso de API, usando una implementación del API UDDI, tal como UDDI4J. Hay siempre 2 diferentes URL's para explorar (lectura de información) y publicar (escritura de información). El ultimo esta usando HTTPS como conexión segura con el fin de asegurarse que la información que publica no esta corrompida o alterada.

Ver ejemplo de registro de negocios en la web (Anexo III Capitulo III Pag. 40 inciso (a))

En general, todos los nodos del registro UDDI tienen la misma funcionalidad, aunque hay diferencias significativas en interfaz de usuario y de uso. Las funciones que son típicamente ejecutadas por los usuarios son:

- **Encontrar negocios, servicios y tModels**

  -Por nombre
  -Por categorización
  -Por clave (key UUID)

- **Navegando a través de negocios, servicios y tModels**

  -Mostrando servicios para negocios
  -Mostrando informaciones obligatorias para servicios
  -Mostrando tModels para los servicios

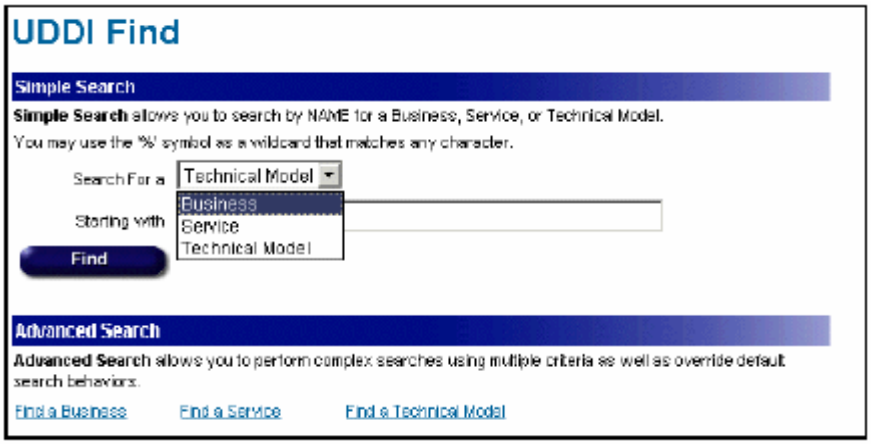
- **Publicando/Cambiando/No-publicando**

  -Entidades de negocio
  -Relaciones del negocio
  -Servicios
  -tModels

1.5.6 ENCONTRANDO INFORMACION DESDE LAS OPCIONES DE BUSQUEDA DE UDDI

Si usted desea encontrar la información construida en el registro UDDI, usted puede elegir entre un conjunto de buscadores, dependiendo de lo que usted este buscando en ese momento, según se muestra en la interfaz de abajo de opciones para búsqueda de los registros UDDI. El acercamiento simple a ello requiere una especificación del tipo de entrada que usted esta buscando y una secuencia que este contenida en el nombre de esa entrada.

Para búsquedas mas avanzadas, utilizar uno de los links llamados find en Business/Service/TechnicalModel. El cual permite uso de la información categorizada y combinaciones de los criterios de búsqueda.



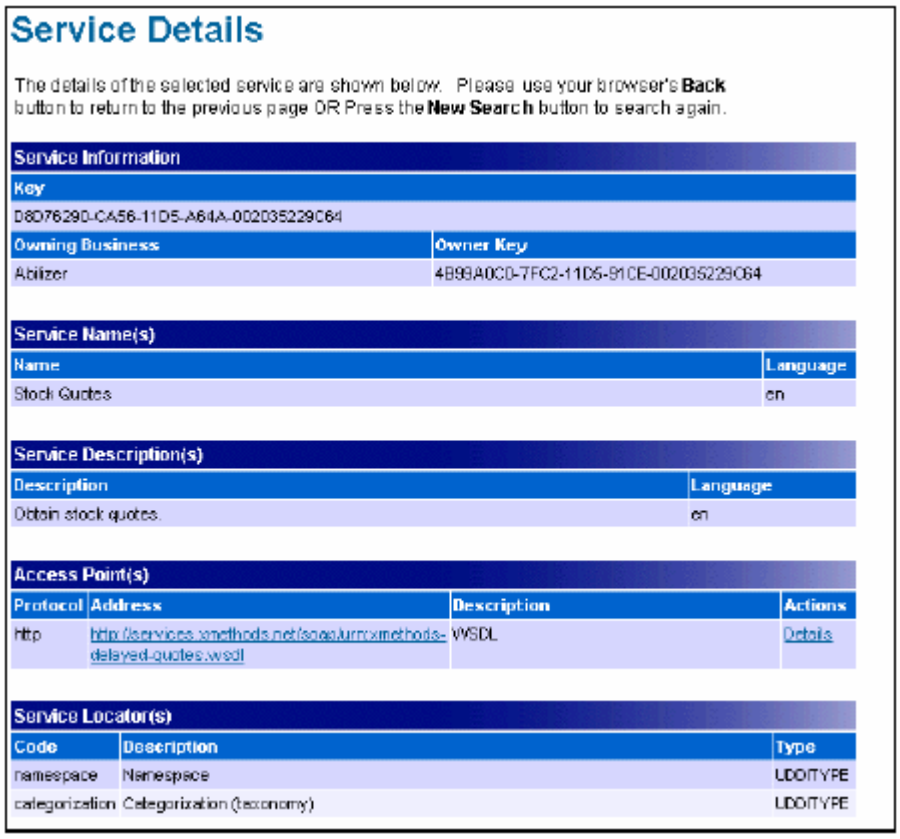
Opciones para búsqueda de los registros UDDI.

Después de que usted tenga un envió exitoso de una consulta de búsqueda, ahora usted podrá navegar a través de los resultados obtenidos, así como desplegar servicios que fueron publicados por un negocio especifico, o mostrar los servicios que implementan una cierta especificación del modelo tModel, etc.

Importante: desafortunadamente hay una gran número de entradas experimentales y de prueba, no solo en el registro de prueba, también en el registro oficial de negocios. Así que cuando se localicé un servicio no se debe intentar llamar un servicio con los puntos de acceso mostrando el nombre del servidor como lo es el localhost, y además aquí no hay garantía sobre la disponibilidad de los servicios.

La figura de abajo muestra un ejemplo de un servicio encontrado en UDDI con la información detallada que puede resultar interesante al cliente que solicita el servicio. Primeramente, aquí es una única ID especificando el servicio, así como también el negocio propietario o que posee el servicio, incluyendo su clave (key). Usted también podría utilizar estas claves (keys) en la característica de búsqueda avanzada.

Nombres y descripciones de servicios, se pueden especificar en distintos idiomas. Aquí se encontraran las entradas para todos los idiomas en la vista de detalle, incluyendo un código de la ISO del lenguaje o idioma usado. Dos piezas esenciales de información son el punto de acceso y el localizador de servicio. El primero es un apuntador a una implementación concreta de este servicio. El último de ellos describe el servicio usando las categorías (taxonomias).



Service Details

The details of the selected service are shown below. Please use your browser's **Back** button to return to the previous page OR Press the **New Search** button to search again.

Service Information

Key	
D8D7629D-CA56-11D5-A64A-002035229C64	
Owning Business	**Owner Key**
Abilizer	4B99A0C0-7FC2-11D5-91CE-002035229C64

Service Name(s)

Name	Language
Stock Quotes	en

Service Description(s)

Description	Language
Obtain stock quotes	en

Access Point(s)

Protocol	Address	Description	Actions
http	http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl	WSDL	Details

Service Locator(s)

Code	Description	Type
namespace	Namespace	UDDITYPE
categorization	Categorization (taxonomy)	UDDITYPE

Explorando detalles del servicio publicado.

El registro UDDI no contiene solo Web Service. Por ejemplo, IBM también publica otros servicios, tales como números telefónicos y URL's para los sitios Web, donde pedir productos de IBM y direcciones de e-mail.

Una vez que se haya identificado un Web Service que se desee utilizar, usted podría seguir el link tModel para descargar el archivo WSDL. Este archivo WSDL incluye todas las descripciones necesarias de la interfaz que se pueden utilizar por herramientas tales como WebSphere Studio Application Developer para desarrollar las aplicaciones del cliente que llaman al Web Service estático. El link del binding template proporciona la URL donde usted invoca el Web Service.

1.5.7 PUBLICANDO INFORMACIÓN DEL NEGOCIO UTILIZANDO LOS REGISTROS UDDI

La segunda tarea que usted podría desear hacer usando registros UDDI es publicar información y servicios de su compañía, de modo que clientes potenciales puedan utilizar sus servicios.

Después de haber seleccionado el registro deseado y se haya firmado exitosamente, usted encontrara un menú adicional en el lado izquierdo llamado publicar (publish). Cliqueando ahí usted vera una ventana con todos los artículos que usted publica en el registro usando su nombre de usuario.

La orden típica en la cual usted publica su información es la siguiente:

1.- Defina un negocio.

Describe su compañía u organización, y opcionalmente asigna una categoría, describe tu dominio del negocio y agrega relaciones a otras entidades de negocios.

2. - Defina tModel (s).

Define los modelos técnicos, refiriéndose a los archivos de WSDL para Web Services o simplemente tModels, tales como mail to o ftp. Note que tModel no se refiere siempre a un archivo WSDL sobre un sitio proveedor. Los tModels contienen una llave única, una descripción y localizadores de clasificación. Si un negocio provee acceso publico a un archivo WSDL, tiene sentido especificar la URL en la descripción del tModel.

3.- Defina servicios y binding

Finalmente se definen sus servicios. Esto consiste de un nombre, clave y alguna información descriptiva. Usted también proporciona inmediatamente la información obligatoria, que en general se compone de un punto de acceso, donde el servicio puede ser activado, y los detalles ligados como son los protocolos usados para conectar (por ejemplo SOAP sobre http). Usted también tiene que referirse a uno o más tModels para describir su interfaz.

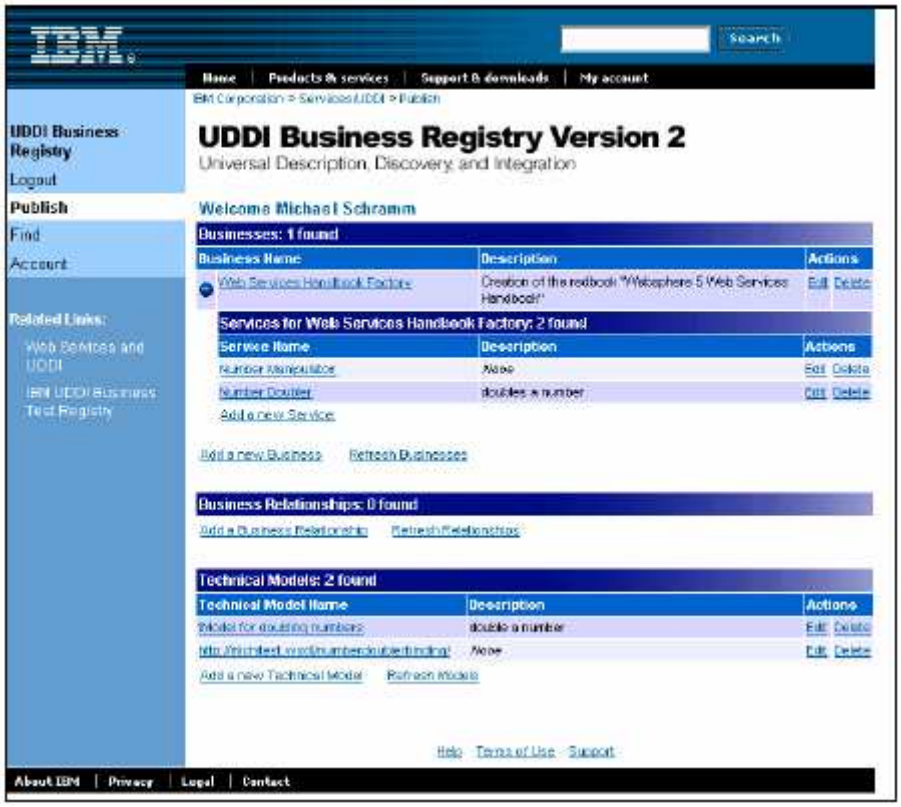
La trayectoria a través de los diversos pasos es absolutamente directa y no se describe detalladamente aquí. Solo hay un hecho interesante a precisar:

- Cuando observamos a la ventana de publicación, usted inmediatamente encontrara su entidad de negocio, relaciones y tModels, incluyendo todos los links para crear unos nuevos. Pero usted no ve inicialmente los servicios del negocio. Esto es porque un servicio (con excepción del tModel) pueden solo existir para una entidad de negocio específica. Tienes que primero expandir una entrada de negocio para ver las definiciones y los links con respecto a los servicios.



Expandiendo entidades de negocio para exploraciones existentes o definir nuevos servicios.

Cuando usted haya finalizado la definición de sus entidades, servicios y tModels, la ventana que usted público se ve como la siguiente:



Descripción de artículos publicados.

PAQUETES IMPORTANTES

- org.uddi4j.datatype

Este paquete contiene clases usadas para enviar o recibir información UDDI.

- Org.uddi4j.request

Este paquete contiene mensajes enviados al servidor. Estas clases son típicamente no usadas directamente, más bien la clase UDDIProxy usa esas clases.

- Org.uddi4j.response

Este paquete representa mensajes de respuesta desde un servidor UDDI.

PRERREQUISITOS

UDDI4J requiere un Java runtime enviroment de la versión 1.2.2 como mínimo y un transporte SOAP, que puede ser uno de los siguientes:

- Apache Axis.
- Apache SOAP 2.3.
- HP SOAP.

USANDO LA LIBRERIA

Usted tiene que realizar algunas preparaciones en orden para usar la librería UDDI. Primero usted tiene que especificar un conjunto de propiedades del sistema donde usted define cual implementación SOAP usted usara y si ahí se encuentran proxies para atravesar.

Si usted no solo desea una exploración al servidor UDDI, si no también desea hacer una publicación de información, usted necesita activar el protocolo de transporte HTTPS para agregar una implementación de JSSE a tu class path. Referirse a la documentación sobre el UDDI4J del sitio Web para información detallada.

1.5.8 ESCRIBIENDO CLIENTES UDDI

Una aplicación cliente típica para el registro UDDI consiste de 3 partes:

- Conectarse al servidor para crear un objeto proxy.
- Encontrar entidades usando este proxy.
- Publicar entidades usando este proxy.

ENCONTRANDO INFORMACIÓN DE SERVICIOS OFRECIDOS POR COMPAÑÍAS

Una vez que usted a creado un objeto proxy, usted puede encontrar fácilmente cualquier tipo de información, tales como entidades de negocio, servicios, bindings, o tModels.

ENCONTRANDO ENTIDADES DE NEGOCIOS

El método find_business es usado para encontrar entidades de negocio por nombre. El método regresa un objeto BusinessList, una colección que puede ser usada para encontrar información especifica acerca de todos los negocios que corresponden a la búsqueda de parámetros.

```
BusinessList UDDIProxy.find_business(
   java.util.Vector names,
   DiscoveryURLs discoveryURLs,
   IdentifierBag identifierBag,
   CategoryBag categoryBag,
   TModelBag tModelBag,
   FindQualifiers findQualifiers,
   int maxRows)
```

Método find_bussiness

Parámetros:

- **Names (Nombres).-** Un vector de nombres. Usted puede usar el carácter % como un comodín.

- **DiscoveryURLs (Descubridor de URL's).-** Esta es una lista de URL's a ser correspondidos o que se emparejan contra los datos asociados con el contenido de los discoveryURLs de la información registrada de la entrada del negocio. El BusinessList regresado contiene las estructuras BusinessInfo que emparejan cualquiera de los URLs pasados (lógico OR).

- **identifierBag.-** Esta es una lista de las referencias del identificador de negocio. El resultado contiene los negocios que emparejan cualquiera de los identificadores pasados (lógico OR).

- **categoryBag.-** Es una lista de referencias de las categorías. El resultado contiene negocios que emparejan a todas las categorías pasadas (lógica AND).

- **tModelBag.-** Los datos registrados de la entidad de negocio contienen binding template que alternadamente contienen referencias específicas del tModel. Los argumentos tModelBag dejan que usted busque para los negocios que tengan binding que sean compatibles con un patrón específico del tModel. El resultado contiene los negocios que emparejan todas las llaves del tModel pasadas. (lógico AND). Los valores dominantes del tModel se deben ajustar a formato como valores cualificados URN de HUID prefijados con el "uuid:".

- **findQualifiers.-** Puede ser usado para alterar el comportamiento por default, la funcionalidad de la búsqueda.

- **MaxRows .-** Permiten que el programa de petición limite el numero de los resultados devueltos.

ENCONTRANDO SERVICIOS

Para encontrar un servicio cualquier criterio de búsqueda, usted debe usar el método find_service:

```
ServiceList UDDIProxy.find_service(
   java.lang.String businessKey,
   java.util.Vector names,
   CategoryBag categoryBag,
   TModelBag tModelBag,
   FindQualifiers findQualifiers,
   int maxRows)
```

Método find_services

Esta función regresa una lista de servicios al ser exitoso. En el caso o evento que no se localicen o situaran para los criterios específicos, la estructura de la lista de servicios regresando contendrá una estructura vacía.

Parámetros:

- **BusinessKey (clave de Negocio).-**Esta opción uuid_key se utiliza para especificar un caso particular de la instancia de la entidad de negocio. Este argumento puede ser utilizado para especificar una

existente entidad de negocio en el registro o podría ser especificada como "" (cadena vacía) para indicar que todas las entidades de negocio deben ser buscadas.

- **Names (Nombres).-** Este vector opcional de nombres representa uno o mas nombres parciales A con carácter comodín % se puede usar para significar cualquier numero de cualquier carácter. Hasta 5 valores de nombre pueden ser especificados. Si múltiples valores de nombres son pasados, la correspondencia ocurre en un lógico OR base dentro de cualquier nombre provisto.

- **categoryBag.-** Esta es una lista de las referencias de la categoría. La lista de servicio regresado contiene las estructuras de BusinessService que corresponden a todas las categorías pasadas (lógica AND por defecto).

- **tModel.-** Esta es una lista de los valores uuid.key del tModel que representa la huella digital de una estructura de un BindingTemplate a encontrar. Si mas de un tModel dominante se especifica en esta estructura, solo la estructura de servicio de negocio que contienen las estructuras de BindingTemplate con la información de la huella digital que corresponde a todas las llaves del tModel especificadas serán regresados (lógico AND solo).

- **findQualifiers.-** Usado para alterar el comportamiento por default de la funcionalidad de la búsqueda.

- **maxRows.-** Permiten que el programa de petición limite el numero de resultados regresados.

ENCONTRANDO tMODELS

Para encontrar tModels, utilice el método find_tModel. Este método es para localizar una lista de entradas del tModel que emparejan un sistema de criterios específicos. La respuesta será una lista de información abreviada sobre los tModels que emparejan los criterios (tModelList).

```
TModelList UDDIProxy.find_tModel(
   java.lang.String name,
   CategoryBag categoryBag,
   IdentifierBag identifierBag,
   FindQualifiers findQualifiers,
   int maxRows)
```

Método find_tModel

Parámetros:

- **name (Nombre).-** Este valor de cadena representa un nombre parcial. Por que los datos del tModel solo tienen un simple nombre, solo un simple nombre puede ser pasado. Un carácter comodín % puede ser usado. El tModelList regresado contiene los elementos tModelInfo para los tModels de los nombres que corresponden al valor pasado.

- **CategoryBag.-** Es una lista de las referencias de la categoría. Los resultados contienen los tModels que corresponden a todas las categorías pasadas (lógico AND por default) FindQualifiers puede ser usado para alterar este comportamiento lógico AND.

- **IdentifierBag**.- Esta es una lista de las referencias del identificador del negocio. El resultado contiene tModels que corresponden a alguno de los identificadores pasados (lógico OR).

- **findQualifiers.-** Usado para alterar el comportamiento por default de la funcionalidad de la búsqueda.

- **MaxRows.-** Permite que el programa de petición limite el numero de resultados devueltos.

Los 3 métodos le permiten explorar negocios, servicios y tModels. Como típico cliente dinámico, usted no esta realmente interesado en el nombre de un negocio o servicio, si no mas en los binding templates que le dicen de donde usted puede acceder al servicio. Un panorama típico seria buscar para un binding usando una llave o clave tModel, extraer el punto de acceso, e invocar el servicio en esa URL.

ENCONTRANDO BINDINGS

Usted puede encontrar bindings usando el método find_binding:

```
BindingDetail UDDIProxy.find_binding(
   FindQualifiers findQualifiers,
   java.lang.String serviceKey,
   TModelBag tModelBag,
   int maxRows)
```

Método find_binding

Este método regresa un objeto BindingDetail que contiene cero o más estructuras BindingTemplate que corresponden al criterio especificado en la lista de argumentos.

Parámetros:

- **findQualifiers.-** Esta colección puede ser usada para alterar el comportamiento por default de la funcionabilidad de búsqueda.

- **ServiceKey (Clave de Servicio).-** Especifica un caso particular de un elemento de Servicio de Negocio en los datos registrados. Solamente los binding en los datos de Servicio de Negocio especificados identificados por la clave del servicio pasado serán buscados.

- **tModelBag.-** Lista de valores del uuid_key del tModel que representa la huella digital técnica para localizar en una estructura de BindingTemplate contenida dentro de la instancia del servicio de negocio especificado por el valor del la clave del servicio. Si mas de un tModel dominante se especifica en esta estructura, solo la información del BindingTemplate que corresponde exactamente todas las claves del tModel especificadas serán devueltas (lógica AND).

- **maxRows.-** Permite que el programa de petición limite el numero de resultados devueltos.

Esta función regresa un objeto BindingDetail en caso de ser exitoso, en el caso que no corresponda será localizado para un especifico criterio. La estructura BindingDetail regresa en la respuesta que será vacía y no contendrá ningún dato BindingTemplate.

En todos los métodos de hallazgo descritos, puedes fijar maxRows de un parámetro para definir el número máximo de resultados. También especificar ahí o en orden para no limitar el resultado. Tener presente que, en el caso de un gran numero de correspondencias, un sitio operador podría truncar el conjunto de resultados. Si esto ocurre, el mensaje de respuesta contendrá el atributo truncado fijado en caso de ser cierto.

PUBLICAR INFORMACIÓN

Para registrar alguna información sobre un servidor UDDI, usar la autorización token obtenida cuando crearon el proxy. El siguiente código muestra como publicar una entidad de negocio.

Este ejemplo solo llena el campo del nombre requerido. De la trayectoria en escenarios reales se podría agregar información de categorización y descripción.

```
// create business entities and store them in a vector
Vector businessEntities = new Vector();
BusinessEntity be = new BusinessEntity("");
be.setName("IBM ITSO");
entities.addElement(be);
// save the entities on the UDDI server
BusinessDetail bd = proxy.save_business
                    (token.getAuthInfoString(), entities);
```

Ejemplo de como registrar información en servidor UDDI

Publicando una entidad de negocio

El objeto BusinessDetail regresado por el método salvar asegura el identificador único (UUID) que se determina para la nueva entidad por el registro UDDI.

Similar a entidades de negocios, también usted puede crear servicios de negocios, y tModels, usando las clases BusinessService o tModel. Cuando quiera usted tener un especifico URLs o punto de acceso, usted también encontrara una envoltura de clases, tal como es OverviewURL o AccessPoint, que prevée un muy fácil e intuitivo camino para modelar sus entradas de registro.

Detalles pueden ser encontrados en http://www.ibm.com/developerworks

1.5.9 REGISTROS PRIVADOS UDDI

La primera implementación de UDDI es el registro de negocios, el cual también algunas veces es llamado la nube (cloud) UDDI. Esta nube consiste de nodos activados por IBM, Microsoft, SAP y otros, que son exactamente copiados o replicados y asegurados con la misma información. Ellos pueden ser accedidos por todos, ambos para explorar información así como publicar información.

Pero en realidad aquí son otros escenarios utilizados que parecen obtener a un más importancia que el registro público, a saber totalmente o parcialmente de los registros privados UDDI. Aquí son un numero de posibles requerimientos que impulsan compañías y organizaciones para establecer esos registros UDDI afuera de la nube.

NECESIDAD PARA PRIVACIDAD

Un requerimiento es que las compañías con frecuencia no deseen mostrar todas sus interfaces a todo el mundo, y por lo tanto también abrir la posibilidad que todos puedan (intentar) comunicarse con sus servicios.

LIBRARSE DE BASURA UDDI

Por que la cloud UDDI es pública y libre para ser usada por todos, es obvio que aquí es frecuentemente inexacto, obsoleto, incorrecto, o hay información falsa en el registro. Aquí no es un concepto de fecha de vencimiento o caducidad para publicar información o un mecanismo de revisión para garantizar la calidad del contenido. Este es un problema similar para el sitio Web del Internet, con la diferencia principal que el usuario del sitio Web son hombres y deben ser capaces de fácilmente separar contenido adecuado o contenido

utilizable de contenido malo. Los clientes de registros UDDI son a menudo aplicaciones. Este hecho puede conducir a problemas severos.

ESTANDAR Y LINEAS DIRECTIVAS

En UDDI usted puede publicar negocios o servicios con muy poca información, y cuando ingresa su URL's para acceder al punto, usted puede especificar archivos WSDL, así como sitio Web, o documentos que describen un servicio. Esto es permitido en UDDI y podría ser exactamente lo que es requerido en algunos casos.

Sin embargo, para hacer automático (aplicación cliente) la exploración (dinámicos Web Services) sencilla, ustedes podrían desear establecer algunas restricciones estandarizar la información que es colocada en algunos lugares específicos del registro.

1.5.10 POSIBLES ESCENARIOS PARA REGISTROS PRIVADOS UDDI

REGISTRO INTERNO

Una compañía u organización podría querer considerar un registro privado totalmente, que reside dentro del cortafuegos (firewall) y puede ser accesible desde dentro de la Internet, ambos programas y base-Web.

Este escenario ayuda a grandes organizaciones para proveer servicios internos a otros departamentos y divisiones, es muy fácil dar la orden a las líneas directivas y restricciones, por que otras compañías no interactúan con un registro.

E-MARKETPLACE DE REGISTROS UDDI

Otro escenario podría ser un registro que es en realidad público, pero especializados en tópicos muy específicos, a un grupo de compañías, o en industrias. Estos registros serán perfectos candidatos para usar taxonomias especializadas, las cuales tienen que ser soportadas desde la versión 2 de UDDI. Por ejemplo, la industria automotriz podría considerarse un establecimiento de registro especializado con su propio sistema de detalles de categorización.

Otra ventaja será el restringir la utilización de algunos conjuntos especiales de tModels. En regla no se permitirá a todas las compañías publicar sus propias interfaces de servicio. La organización que almacena (hosting) los registros podrían querer considerar definiciones de estandarización tModels y WSDL que son soportados por este registro.

Usando estos registros especializados incrementa mucho la posibilidad de automáticamente descubrir e invocar Web Services para aplicaciones de clientes.

REGISTROS UDDI EN EXTRANET

Un tercer escenario es usado por UDDI donde una compañía u organización registra un hosts para la comunicación entre la compañía propietaria y el negocio asociado. Así que el registro esta sobre la Internet, pero el acceso es restringido a la compañía propietaria y el socio con previa negociación de contrato entre los negocios.

Esta permitido un buen compromiso entra la privacidad de una compañía y la habilidad o capacidad para comunicarse fácilmente con los socios. Todas las ventajas de registros privados aplican, manteniendo el registro limpio y uniforme.

1.5.11 BENEFICIOS DE REGISTROS PRIVADOS UDDI

La utilización de escenarios de los registros UDDI no son restringidos a los registros de negocios públicos. Aquí hay muchas razones por que una organización puede considerar establecer un registro que no es parte de los registros oficiales.

Para hacer esto posible restringirá una o más de los siguientes puntos:

- Quien es permitido para explorar el registro.
- Quien puede publicar información.
- Que tipo de información es publicada (líneas directivas, estándar)

Otro importante punto acerca de los registros privados es que el porcentaje de éxito para exploración dinámica ejecutada por la aplicación cliente se incrementa dramáticamente.

CONSIDERACIONES ADICIONALES PARA REGISTROS PRIVADOS UDDI

Dos consideraciones cuando piense en algún registro que no es parte del registro central del negocio.

GARANTIZAR APIS

En cada caso se puede considerar el permitir o deshabilitar la exploración y publicación de API's.

POLÍTICAS

Aquí están muchos escenarios diferentes en los cuales un registro puede ser usado:

- Publico en el Internet.
- Parte publica en la extranet.
- Privado dentro de la intranet.
- Solo partes pequeñas para desarrollo.
- Solo estables para pruebas aéreas.
- Solo escalables para ambientes de producción.

Cuando uno de estos registros requieren ligeras diferencias de comportamiento, puesto que esto pretenderá usarlo, así aquí están un conjunto de políticas que pueden ser aplicadas al registro que cambiara el actual comportamiento respecto a:

- Autorización de modelos.
- Datos cuidados y confidenciales.
- Generación de claves.
- Valor fijado de validación.
- Suscripción.

- Usuario de publicación limitado.
- Política de auditoria.

1.5.12 RESUMEN UDDI

UDDI es un estándar que provee la habilidad de usar dinámicos Web Services, que significa que los servicios solicitados y servicios provistos no son necesariamente conocidos uno acerca del otro. Una organización que desea proveer algún servicio publicando en algún registro UDDI.

Aquí están dos caminos para que un cliente pueda encontrar un Web Service:

- Realizando una exploración personal de la UDDI en tiempo de diseño, el que podría buscar por servicio o un WSDL y usar esa información cuando programa el cliente.
- Otra posibilidad será un programa de exploración por la aplicación cliente, el cual permite dinámicos binding y cambios de servicios suministrados en tiempo de ejecución.

Se presentan 3 puntos principales que son a menudo considerados o no entendidos cuando hablamos acerca de registros UDDI.

Un registro no solo es un Web Services con accesibilidad directa de archivos WSDL publicados, también alguna otra categoría de servicio, la cual no puede ser usada para programar.

Es un incremento necesario para especializar y dividir registros privados que son hospedados por organismos estándar o industrias. En suma registros internos en intranet sin algún contacto a otros negocios asociados o clientes están ganando más y más importancia.

Cuando publicamos o encontramos tModels, usted nunca publicara o cargara archivos de interfase WSDL. Los tModel son solo una liga lógica a una interfase concreta. Cuando un servicio se refiere a un tModel, usted no puede directamente encontrar fuera la interfase real.

II UTILIZACIÓN DE SOFTWARE COMERCIAL (WEBSPHERE).

Se trata del primer servidor de aplicaciones que ofrece capacidad de detección y recuperación simultáneas para proteger los sistemas críticos, un nuevo software diseñado para proteger las aplicaciones de negocio de Internet de posibles fallos en el sistema o en el suministro eléctrico. Estas incidencias pueden llegar a generar a las compañías pérdidas de hasta 90.000 euros por minuto.

La versión 6 de WebSphere Application Server ha sido diseñada para detectar automáticamente los problemas - ya sean pequeños fallos de red, cortes de suministro eléctrico o desastres naturales - y, en cuestión de segundos, almacenar y procesar las transacciones de negocio basadas en la web, una labor que podría llevar horas o incluso días con los sistemas antiguos. Hasta ahora, las compañías dependían de procesos manuales o tenían que dedicar tiempo y recursos para vincular diferentes paquetes de software, de tal forma que pudieran proteger y recuperar la información.

WebSphere Application Server v6 ayuda a proteger las aplicaciones de negocio basadas en Internet gracias a sus capacidades avanzadas de informática autónoma. Esta tecnología permite generar infraestructuras de tecnologías de la información que pueden configurarse, repararse, optimizarse y protegerse a sí mismas. Así, las compañías pueden centrar sus recursos en las necesidades de su negocio.

Después de detectar una avería en el sistema de suministro eléctrico, WebSphere Application Server v6 de IBM redirecciona automáticamente todos los datos a un servidor paralelo que actúa como sustituto del principal en caso de producirse cualquier fallo. Este servidor puede localizarse dentro del mismo centro de proceso de datos o, en caso de tratarse de una avería más seria o un desastre natural, WebSphere Application Server puede transferir la información vía Internet a una ubicación remota.

Soporte de estándares de servicios web. WebSphere Application Server v6 soporta todos estos estándares, incluyendo WS-Segurity, que autentica las comunicaciones entre servicios web, y WS-Transactions, diseñado para asegurar que las transacciones de servicios web funcionan consistentemente. WebSphere Application Server también soporta 30 plataformas diferentes de sistemas operativos, más que ninguna otra solución de la industria.

Toda la familia de soluciones WebSphere Application Server, incluyen soporte para Java 1.4, que facilita el desarrollo e implantación de aplicaciones utilizando herramientas estándar de la industria, y el soporte para los últimos estándares de servicios web, lo que permite la integración de aplicaciones tanto interna como externamente con clientes, socios y proveedores.

Construcción y desarrollo de servicios web

La nueva configuración de WebSphere Aplication Server y WebSphere Studio ayudará a los desarrolladores a construir y desplegar nuevas y avanzadas aplicaciones para servicios web, ofreciendo a los clientes capacidad para llevar a cabo las siguientes funciones:

- Crear, entre otras, nuevas aplicaciones combinando de forma lógica servicios web existentes y aplicaciones basadas en J2EE. Las nuevas herramientas de WebSphere Studio ofrecen a los desarrolladores una vía intuitiva para visualizar aplicaciones interrelacionadas a través de un simple interfaz de arrastrar y soltar (drag-and-drop), además de la posibilidad de reutilizar dichas aplicaciones como servicios web.

- Utilizar la Arquitectura de Conector J2EE (JCA) para unir nuevas aplicaciones basadas en Java con aquellas no basadas en este estándar. La nueva tecnología de servicios web de WebSphere incluye herramientas de desarrollo de fácil uso para crear sofisticados adaptadores para conectores JCA,

añadiendo un completo soporte transaccional, comunicación entre aplicaciones e incremento en la fiabilidad.

- Incorporar flexibilidad, permitiendo a los servicios web y a las aplicaciones basadas en J2EE adaptarse a las necesidades de negocio. Separando las reglas del negocio de la lógica principal de las aplicaciones, éstas pueden ser modificadas para reflejar las cambiantes condiciones de comercialización -como nuevos descuentos ampliados a socios comerciales- sin tener que parar y restaurar la aplicación. Esto aumenta notablemente la eficacia y la capacidad de respuesta de una empresa. Las aplicaciones también pueden ser dinámicamente adaptadas a usuarios internacionales, reflejando parámetros como el tiempo y el tipo de cambio.
- Proporcionar a las empresas una infraestructura completa para implantar internamente la tecnología de servicios web y conseguir una mejor integración con clientes y socios de negocio. El registro WebSphere UDDI, integrado dentro del software de infraestructura WebSphere, permite a las empresas buscar y agregar recursos a través de diferentes aplicaciones y divisiones y poder así desarrollar aplicaciones de servicios web que se puedan utilizar tanto dentro como fuera de la organización.

III. IMPLEMENTACION DE UN WEBSERVICE UTILIZANDO JBOSS.

Contemplar en este capítulo el uso de el servidor de aplicación por parte del grupo de desarrollo Open Source JBoss el cual esta basada su arquitectura en la plataforma Java 2 Enterprise Edition (J2EE) y a su vez contiene un servidor web Tomcat o Jetty .

Para estudiar la funcionalidad y los alcances de un WebService utilizar la aplicación Duke´s Bank la cual es tomada de la empresa Sun Microsystems y desplegada esta misma a una versión de JBoss. El soporte lo proporciona la organización Apache con el conjunto de herramientas esenciales: Ant y Axis, la primera es utilizada para el desarrollo del proyecto y la segunda es el soporte para WebService.

3.1 REQUERIMIENTOS Y SOFTWARE A UTILIZAR

El sistema operativo a utilizar es Linux Red Hat 8.0, el WebService es implementado e invocado en la misma máquina local.Cabe mencionar ,que la implementación se ha de realizar a su vez en Windows XP para efectos de pruebas, es decir, la idea es tener un conjunto de clientes WebServices capaces de interoperar y utilizar tanto envio como recepción de datos a través de protocolo SOAP y su conjunto de librerías asociado.. Por tanto, este programa cliente reccae por parte de un desarrollo en Visual Basic Studio .Net propiedadde Microsoft. Debido a lo anterior y que la aplicación corre en forma local, es importante instalar el servidor JBoss y el conjunto de programas listados a continuación para establecer una prueba más contundente de cómo un programa cliente WebService realizado en una plataforma distinta puede comunicarse con una aplicación basada en plataforma J2EE.
La lista de software libre a utilizar son :

Máquina virtual java (JDK versión 4.0)
Una versión de Ant (Apache-Ant versión 6.0.2.)por parte del grupo Apache, como herramienta dedespliehue de la alicación..
Apache-Axis versión 1.1
Tutorial J2EE versión 1.4 Sun Microsystems
Servidor JBoss versión 3.2.2 implementación de JBoss-Net (WebServices)
Servidor JBoss versión 4.0.1 implementación de JBossWS (WebServices)
Apache Struts

Hardware
IBM Netvista Procesador Intel IV
Memoria 386MB en RAM

Apache Axis representa el componente de software libre que hace posible utilizar la tecnología Webservice e implementarla en conjunto con el servidor de aplicación para exponer los componentes J2EE así como la creación de algún servicio en particular siempre que éste desee utilizar el protocolo SOAP. Éste mismo es un servidor y cliente de licencia libre SOAP , un mecanismo para la comunicación de las aplicaciones entre sistemas escritos en lenguajes arbitrarios a través de Internet.

Axis implementa técnicamente el API JAX-RPC, uno de los estándares para programar servicios en Java.

Axis es compilado en un archivo JAR (axis, jar), implementa JAX-RPC declarado en los archivos jaxrpc.jar y saaj.jar.

3.2 APLICACIÓN DE SERVIDOR JBOSS

Obtener las versiones de los servidores de aplicación de JBoss es muy simplemente solo hay que bajar ya sea los archivos binarios o con el código fuente; los archivos viene en extensión .zip, tar.gz, bz2. Se puede seleccionar de acuerdo a la plataforma correspondiente principalmente distribuciones de Unix o Windows.

Los requerimientos de instalación son simplemente una versión JDK

JBoss cuenta con una vista del servidor llamada consola de aplicación JMX (JBoss Management Console) donde posee información del conjunto de Mbeans los cuales constituyen el servidor, es decir, el conjunto de servicios por el cual esta integrado, por ejemplo service=JNDIview el cual proporciona una la vista a la estructura de los nombres de espacio JNDI dentro del servidor.; también compuesto por un microkernel . Con esto hace posible obtener un servidor modular y flexible donde puede adquirir diversas configuraciones donde también se puede remover componentes que no se requieren. Dichas configuraciones que son: all, default y minimal cada una de ellas maneja una cierta cantida de servicios que serán inicializados.

JBoss posee en sus versiones componentes diseñados para implementar un Web Service

Por ejempo:

JBoss-Net

Es el modulo donde contiene y provee Web Services contruido por Apache Axis SOAP y la intención es integralo J2EE y JMX. Este incluye un tipo de archivo Web Service Archive (WSR) el cual permite desplegar y empaquetar Web Service dentro de JBOSS esta incluido este servicio solo en la configuración all de la versión 3.x.x.

JBoss WS – Jboss Wiki

Modulo responsable de proveer WebService reemplazando el paquete JBoss-Net es una versión mejorada y aumentada pero de igual forma esta basado como su precesor en Apache – Axis (http://ws.apache.org/axis) este contiene el conjunto completo de componentes de J2EE 1.4 para WebServices, incluyendo SOAP, SAAJ, JAX-RPC, y JAXR esta contenido en las versiones JBoss 4.x.x

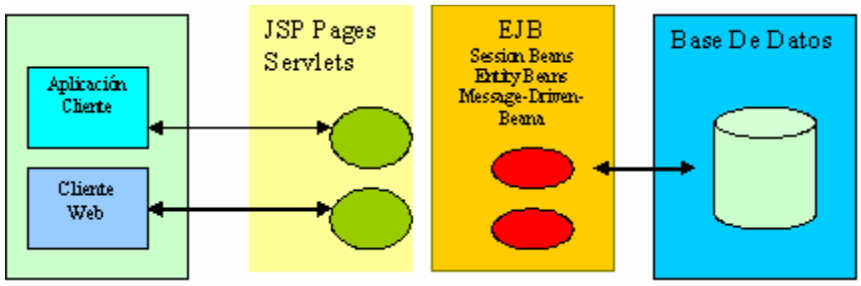
3.3 LA ARQUITECTURA J2EE

Arquitectura multi-funcional para implementar aplicaciones enterpise-class y aplicaciones basadas en web . La tecnología soporta una gran variedad de aplicaciones desde gran escala a cliente-servidor. El principal objetivo de J2EE es crear un modelo de desarrollo simple para aplicaciones empresariales utilizando componentes basados en en un módulo de aplicación. En este modelo tales componentes usan servicios que provee el contenedor, el cual necesita ser incorporado en un código de aplicación

Componentes:

Un componente J2EE es una unidad funcional de software que se ensambla en la aplicación J2EE con sus clases de ayuda y archivo que se comunican con otros

Aplicaciones cliente, applets que corren en el cliente
Java Servlets y Java Servlets Pages (JSP) que son componentes Web que corren en el lado del servidor
Enterprise Java Beans que son componentes de negocio que corren en el servidor de aplicación.



Entorno Cliente	Servidor J2EE	Servidor DB

Principales componentes de una arquitectura J2EE

En suma a esos componentes incluye servicios estándar y tecnologías las cuales son:

Java Database Conectivity (JDBC) que provee el sistema de base de datos relacional.

Java Transacion API (JTA) Java Transaction Service(JTS) que provee transacciones de apoyo para módulos J2EE.

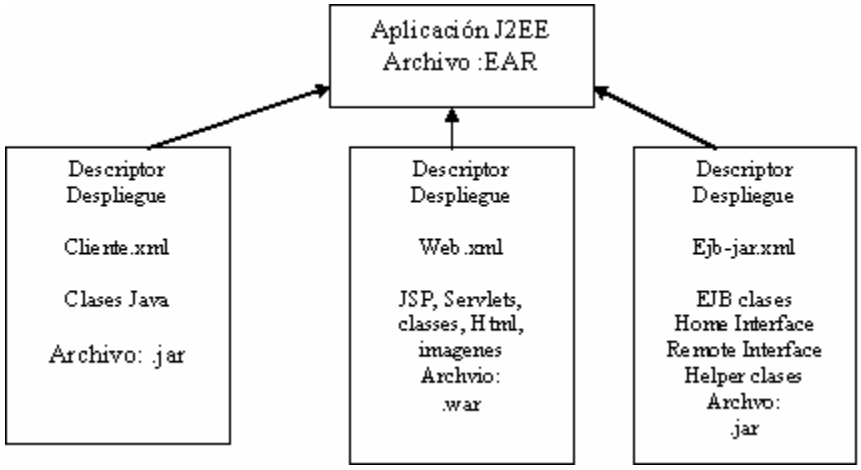
Java Naming Service(JMS) para comunicaciones asíncronas entre componentes J2EE.

Java Naming and Directory Interface(JNDI) que provee nombre y directorio de acceso empaquetamiento

Descriptores de despliegue. Contiene información específica de cada componente y su comportamiento (deployment descriptors)

WAR(Web Archive)	Componentes Web
JAR (Java Archive)	Componentes de Negocio
JAR (Java Archve)	Cliente
EAR(Enterpsie Archive)	Contiene la aplicación J2EE en conjunto

Descriptores de despliegue



Esquema componentes J2EE y descriptores de despliegue

3.3.1 EJB

Un enterprise java bean es un componente del servidor que encapsula la lógica del negocio en la aplicación.

Permiten simplificar el desarrollo de grandes y distribuidas aplicaciones. Primero porque el contenedor EJB provee varios servicios de sistema para los enterprise beans, el desarrollador de los beans solo se concentra en resolver problemas del negocio. El contenedor EJB es responsable para los niveles de servicio del sistema tales como la administración de las transacciones y autorización segura.

Los beans contiene la lógica de negocio así que el desarrollador se concentra en la presentación del cliente. El desarrollador del cliente no tiene que programar las rutinas que implementa la lógica del negocio de acceso a la base de datos. Por lo que los clientes corren en pequeños dispositivos.

Los enterprise beans son componentes portables, el ensamblador de la aplicación puede construir nuevas aplicaciones desde la existencia de los beans estos corren en cualquier Servidor J2EE.

Para usar un enterprise Java bean es importante que la aplicación:

- Sea escalable, que permita acomodar un gran número de usuarios, no se requiere distribuir la aplicación en múltiples máquinas y su localización es transparente a los clientes.

- Las transacciones son requeridas para asegurar la integridad de los datos. Enterprise beans soportan transacciones, el mecanismo que maneja el acceso actual de objetos compartidos.

Tipos de Enterprise Java Beans

Tipos de Enterprise Beans	Propósito
Session	Ejecuta tareas para el cliente
Entity	Representa un objeto de entidad de negocio que existe en el almacenamiento persistente.
Message-Driven	Actúa como receptor para el API Servicios de Mensaje Java Procesa mensajes asincronamente

Tipos de enterprise Java Beans

Session Bean

Representa un solo cliente dentro del servidor J2EE. Para acceder a una aplicación que es desplegada en el servidor, el cliente invoca los beans del método. El session bean ejecuta el trabajo para sus clientes para realizar tareas de negocios dentro del servidor, en pocas palabras es un sesión interactiva por lo que los datos no son persistentes (los datos no son guardados a la base de datos).

Existen dos tipos de session Bean:

Stateful session Bean: El estado del objeto consiste de los valores de sus instancias variables, esto es, que representa el estado de un único cliente en la sesión. Por que el cliente interactúa con el bean , este se llama estado conversacional. El estado es retenido por la duración de la sesión del cliente si el cliuentye remueve el bean o termina, la session finaliza y el estado desaparece.

Stateless Session Bean: El bean no mantiene un estado de conversación para un cliente en particular. Cuando el cliente invoca el método de un stateless bean, la variable de instancia del bean puede contener un estado, pero solo por la duración de la invocación. Cuando el método finaliza, el estado no es retenido más. Excepto por el método de la invocación, todas las instancias de un stateless bean son equivalentes, permitiendo que el contenedor EJB asigne una instancia para cualquier cliente.
A la vez, el contenedor EJB puede escribir un stateful session bean para un almacenamiento secundario. Sin embargo, el stateless session bean nunca son escritos a un almacenamiento secundario. Por ello, stateless beans pueden ofrecer un mejor comportamiento que statelessful beans.

El estado del bean no contiene datos para un cliente específico. La invocación de un solo método, el bean ejecuta una tarea genérica para todos los clientes. Del bean toma desde la base de datos un conjunto de datos de lectura y escritura utilizados por el cliente.

Entity Bean

Representa el objeto en el negocio en un mecanismo de persistencia y esta es una de las características mas, permite un acceso compartido, tener llaves primarias, y participa en la relación con otros entity beans.

La Persistencia significa que el entity bean existe mas allá de sus ciclo de vida de la aplicación o del servidor del proceso de servidor J2EE.La persistencia esta relacionada al concepto en base de datos de que la información existe aun después de apagar el servidor de base de datos o a las aplicaciones que sirve.
Los dos tipos de persistencia para entity beans son: bean-managed-persistence, donde el código del entity bean que se escribe contiene las llamadas que acceden a la base de datos. Si el entity bean tiene container-managed-persistence, el contenedor EJB automáticamente genera las llamadas de acceso a la base de datos.
Entity beans pueden compartir múltiples clientes. Debido a que los clientes pueden compartir el mismo dato.
También cada entiry bean tiene un objeto identificador único. Un ejemplo de cliente entity bean , podría ser identificado por un número . El identificador único o llave primaria, habilita al cliente para localizar un entity en particular.

Message-Driven Bean

Procesa mensajes asincrónicamente los mensajes pueden ser enviados a cualquier componente J2EE y aplicación cliente, a otro enterprise bean o a un componente Web o por medio de una aplicación JMS (Java Message Service) o sistema que no use la tecnología J2EE.

La diferencia de los dos beans anteriores es que message-driven-beans no tiene acceso por parte de los clientes, solo contiene una clase.

3.4 ASPECTOS DE LA APLICACIÓN DUKES BANK

La aplicación esta basada en un simple banco en línea el cual cuenta con dos clientes: un cliente Java para administrar el manejo de cuentas y clientes, y un cliente Web exclusivo para los clientes que desean acceder a sus estado de cuenta y realizar ciertas transacciones. El funcionamiento de la aplicación es poder tener cierta información de las cuentas, tanto el número, tipo de cuenta, saldo, etc.; así también, contar con estado de cuenta de acuerdo a la fecha solicitada; realizar operaciones como son depósitos, traspasos,retiros, entre otros.

Esta misma esta compuesta por la tecnología enterprise beans, así como componentes Web. Servlets, JSP.

Descripción breve de la Funcionalidad de la aplicación Duke`s Bank

La aplicación basada en tecnología J2EE reúne los componentes como es en primera instancia un cliente web y también un cliente java; del lado del servidor se encuentra el contenedor web con el conjunto de servlets y jsp: **Account List**, **AccountHistory**, **Transferfunds**, **ATM**; un contenedor EJB compuesto por **AccountController Bean**, **CustomerController Bean** y **TxControllerBean**.Cualquiera de ellos sirve como prueba para exponerlo como un Web Service.

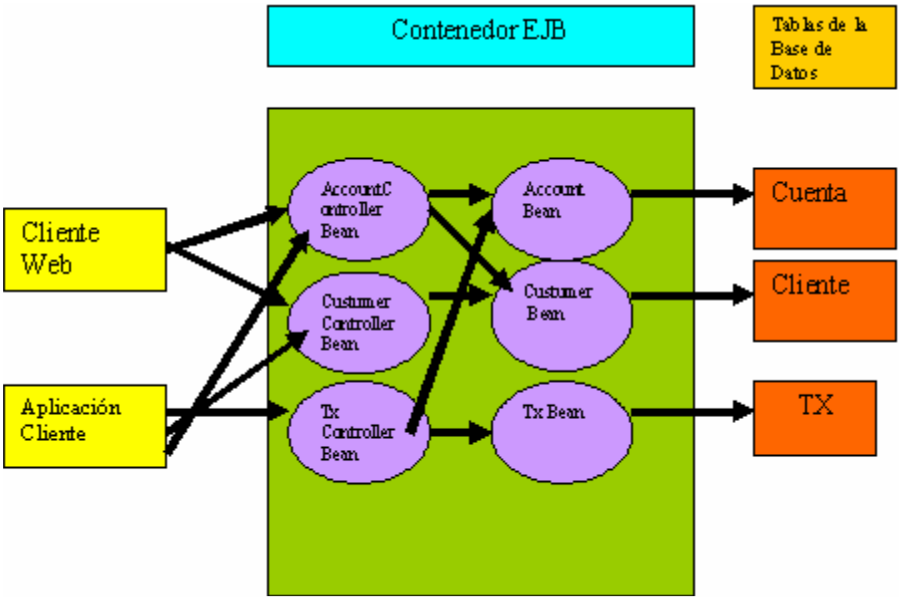
La función de AccountController es crear y mover entity beans los cuales manejan la relación cuenta-cliente y por consiguiente su función es recoger información de la cuenta.

Los métodos dentro de AccountController que manejan la cuenta son:
- **getAccountsOfCustomer**
- **getDetails**.

CustomerController Bean es encargado de dar de alta clientes nuevos, dar baja de estos a través de sus métodos correspondientes.

TxConytroller Bean maneja las transacciones bancarias como son retiros, depósitos, cargos en cuentas, pagos y transferencia de fondos. Con los métodos correspondientes:
- **Retiro**
- **Deposito**
- **Hacercargo**
- **Hacerpago**
- **TransferenciadeFondos**

Estos métodos antes tienen que entrar a AccountController Bean para verificar tanto el número de la cuenta con su cliente respectivo.



Esquema de componentes de la aplicación Dukes Bank

La base de datos esta conformada por Hypersonic la cual se encuentra embebida en el servidor JBoss, aunque existe la posibilidad de utilizar otra base de datos más poderosa y versátil como es Postgress o bien My SQL.

Al revisar la aplicación que del banco virtual la pantalla principal se despliega en la url: **http:/localhost:8080/bank/main**[1], siendo ésta la interfas de el cliente cliente web. Es un JSP el cual pide la autenticación, entregar número de cliente y el password respectivo.

Las siguientes figuras muestran los JSP de la aplicación corriendo baja el servidor JBoss y el comportamiento adecuado sus EJB lo que demuestra que esta lista para utilizar uno de sus componentes como modelo de negocio, en otras palabras la empresas pueden montar cualquier aplicación y aplicar esta misma para que existan clientes que puedan interactuar pero a través de sus propios sistemas o aplicaciones, es decir, comunicarse directamente enviando peticiones en particular donde se requiere la respuesta que resida en la misma aplicación.

Teniendo esta aplicación simple basada en la arquitectura J2EE es posible tomar componentes de ésta misma para exponer ymodificar él código correspondiente para iniciar un WebService que en este caso es pisbile tener a un EJB que es la parte de la logica del negocio que puee brindar un comportamiento y manipulación orientada a clientes

3.5 DESPLEGANDO WEBSERVICE DE LA APLICACIÓN DUKESBANK

Teniendo la página WSDL como una interfase donde describe el propósito del negocio, los métodos que se incluyen, el tipo de datos que se pueden pedir y las respuesta a estos; permite a otros clientes diseñar a través de cualquier lenguaje de programación o plataforma un medio de acceso directo a los recursos que puede ser por ejemplo solicitar información de servicios o productos , hacer peticiones que son importantes que lleguen a los sistemas y puedan ser utilizados estos mismo de acuerdo a las necesidades de los clientes y de sus aplicaciones. Pero para ello es importante señalar que cada lenguaje de programación cuenta con paquetes ya sea **Visual Studio .Net**, **SOAP: Lite para Perl**, **Nusoap para PHP**, que son el conjunto de librerías SOAP

[1] **Ver anexo 5**

capaces de ayudar al desarrollo de aplicaciones o programas cliente para Web Services. De esta forma tener en mente que la base de la tecnología de los Web Services es a través del protocolo SOAP (Simple Object Access Protocol) que son un conjunto de mensajes en lenguaje XML o bien un estándar en la interoperabilidad.

El proyecto parte de probar, esos pequeños o simples componentes que puedan ser accesados por varios clientes creados de diversos lenguajes y plataformas utilizando líbrerías de SOAP, y una de opción es tomar un EJB (Enterprise Java Beans) de la aplicación, aunque también puede ser un JSP o un Servlet, por lo cual es más factible montar un Stateless Session Bean debido a sus características y funcionalidades en la interacción con los clientes.

Los Web Services en J2EE provee **endpoints**, llámese también, el lugar donde se procese las peticiones de los clientes y se tengan su respuesta ahí misma; comúnmente si se piensa exponer un Web Service como una capa independiente de la plataforma, entonces el **endpoint** funciona como un objeto que expone operaciones de una invocación solo dentro de ésta

Teniendo el endpoint que requiere el Web Service es importante crear la totalidad del servicio partiendo de una página WSDL, la finalidad es exponer la descripción de nuestro servicio específicamente el conjunto de métodos y datos que pueden ser aprovechados de la aplicación, en este caso Duke´s Bank, a cualquier cliente. Para que posteriormente sean publicados los recursos de éste sistema ya sea en UDII o en Eb-XML

Para demostrar la funcionalidad y comportamiento de esta tecnología es por medio de la invocación del servicio y para ello es importante no solo crear un cliente java que interactué y solicite y reciba datos de la aplicación sino establecer la comunicación entre otros clientes escritos en otra plataforma, otros lenguaje de programación y un ejemplo factible es utilizar Visual Studio .Net de Microsoft. El uso de esta herramienta no es para explicar en ella el funcionamiento de lo Web Service sino un medio en el cual se invoque los servicios de la aplicación, comprobar la comunicación haciendo uso un cliente en Visual Studio una arquitectura y el uso de un lenguaje distinto a J2EE con el lenguaje de programación java.

Por tanto lo mas importante para probar el Web Service es primero establecer la instalación de nuestro servidor de aplicación a cargo de JBoss. Para ello el proceso no tiene ninguna complicación es simplemente bajar a través del sitio correspondiente las versiones binarias e instalarlas. Cabe señalar, que al momento de correr èl mismo, verificar que no se presente ningún tipo de error que puedan perjudicar en el funcionamiento de los servicios[2]

La configuración utilizada debe incluir el servicio de WebService en este caso o bien JBossWS, así como la base de datos (Hypersonic) par el correcto funcionamiento de la aplicación, de esta forma el grupo JBoss se encuentra soportado por herramientas de **Apache org** la cual ha de proporcionar al servvidor Axis como el soporte para la tecnología de Web Service.

La herramienta que permite el desarrollo de los procesos **Apache Ant**: creación, prueba, y despliegue de proyectos java. Esto es para lograr la eficiencia en la aceleración de ciertas tareas Compilación, crear y remover carpetas y archivos, creación de archivos jar tar, zip ; archivos (enterprise application archive) .ear, así como incluir procesos independientes como automatizar ciertos comandos, posee tareas como WAR donde produce la creación y despliegue de componentes web en el archivos war (web archive) y ejb-task para el despliegue y desarrollo de EJB Éste mismo puede integrarse otras herramientas IDE (JEdit, Forte, Eclipse) que también es un esquema que permite el desarrollo de aplicaciones pero con interfases gráficas, muchas de ellas incluyen ANT como parte de herramienta de desarrollo, la cual permite simplificar procesos de creación de archivos, ejecución, despliegue , compilación, depuración, etc.

ANT posee la facultad de ser multiplataforma, pero tiene la opción de hacer de sus archivos build o scripts trabajen solo en una plataforma específica.

---

[2] **ver anexo 7. Instalación de la aplicación Duke´s Bank en el servidor de aplicación JBoss**

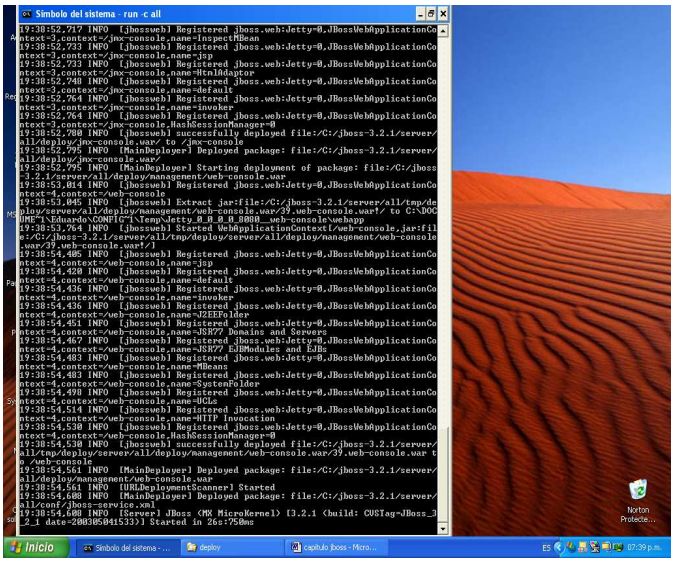
El servidor JBoss se ha de inicializar con el srcipt run.bat o run.sh de acuerdo a la plataforma en que este instalado ya sea Windows o UNIX.

Recordando que para utilizar JBOSS-NET el servidor JBoss-3.2.1 requiere la configuración all mientras que JBoss-4.0.1 solo requiere solo default. Para JBossWS.

El servidor envía una serie de mensajes donde esta verificando si los servicios se han restaurado correctamente (INFO) así también mensajes cuando existen alguna falla (ERROR o WARN) que consideraciones que hay que tomar en cuenta para que la aplicaciones montar no tenga problemas de despliegue.

Al final existe el mensaje que el microkernel del servidor se ha desplegado y el timpo en que ha tardado su instalación.

Ahora sí los servicios están disponibles



Inicio del servidor Jboss versión 3.2.1 en WindowsXP

Es momento de desplegar la aplicación, ANT proporciona las tareas requeridas para su implementación en JBoss.

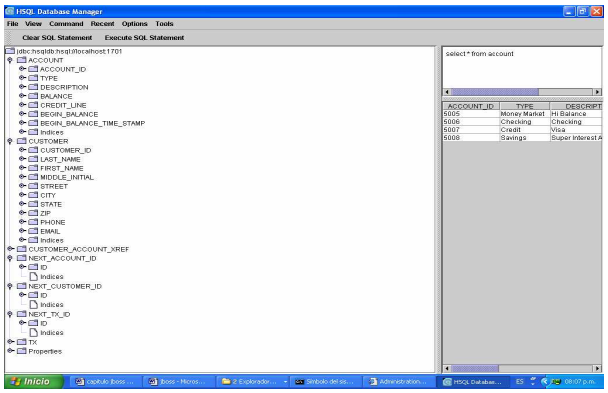
Posterior a la intalación y verificación la estabilidad del servidor donde no presente mensajes de error, es desempaquetar el Tutorial J2EE versión 1.4, este mismo contiene una estructura de directorio donde incluye la carpeta src donde se encuentra el código fuente de la aplicación bancaria; build : donde se depositarán todas las clases y componentes ya compilados; dd: aquí se encuentra los descriptores de despliegue para los EJB(ejb-jar.xml, jboss.xml) , web (jboss-web.xml y web.xml) y cliente(application-client y jboss-client.xml y jndi.properties); sql: los scripts que contienen la creación de las tablas para la base de datos create table.sql y para el llenado de la misma insert-table;

El archivo **jboss-build**[3] como script utilizado por ANT contiene la declaración de los path y librerías ha utilizar, las tareas que ANT realiza son para que el proyecto sea desarrollado y desplegado en JBoss de una forma automatica simplificando los procesos.

La aplicación Dukes Bank en JBoss, resulta de forma sencilla por medio de ANT, cuando ya se tiene precompilado el código y empaquetado en archivos **.jar** y **.war** que finalmente para el despliegue total es conveniente utilizar el empaquetamiento en un archivo **.ear**. El despliegue reside en la carpeta JBOSS-HOME/default/deploy

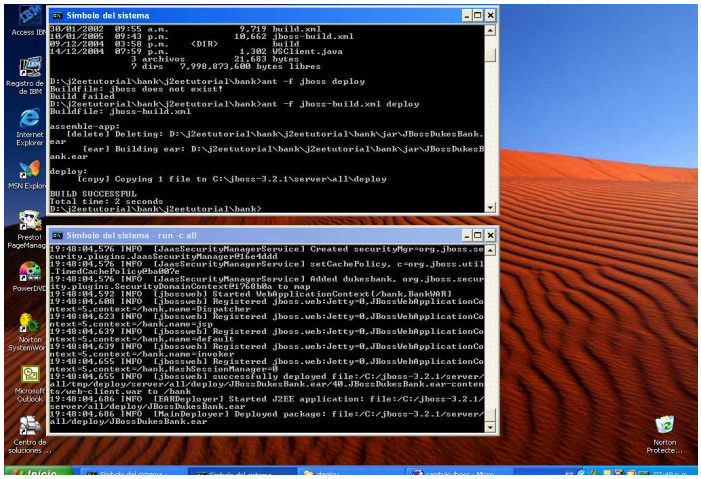
Existe un mensaje en pantalla cuando se tiene éxito y reconoce el servidor los componentes de la aplicación por medio del archivo JBossDukesBank.ear que es así como se ha distinguido

Antes de ello ANT permite llenar la base de datos haciendo uso de los scripts que se encuentran en la carpeta SQL del Tutioral J2EE. Esta base es Hipersonic que se encuentra embebida en el servidor JBOSS incluye un administrador de la base de datos que puede ingresarse como se ve en la figura:



Administración Hipersonic

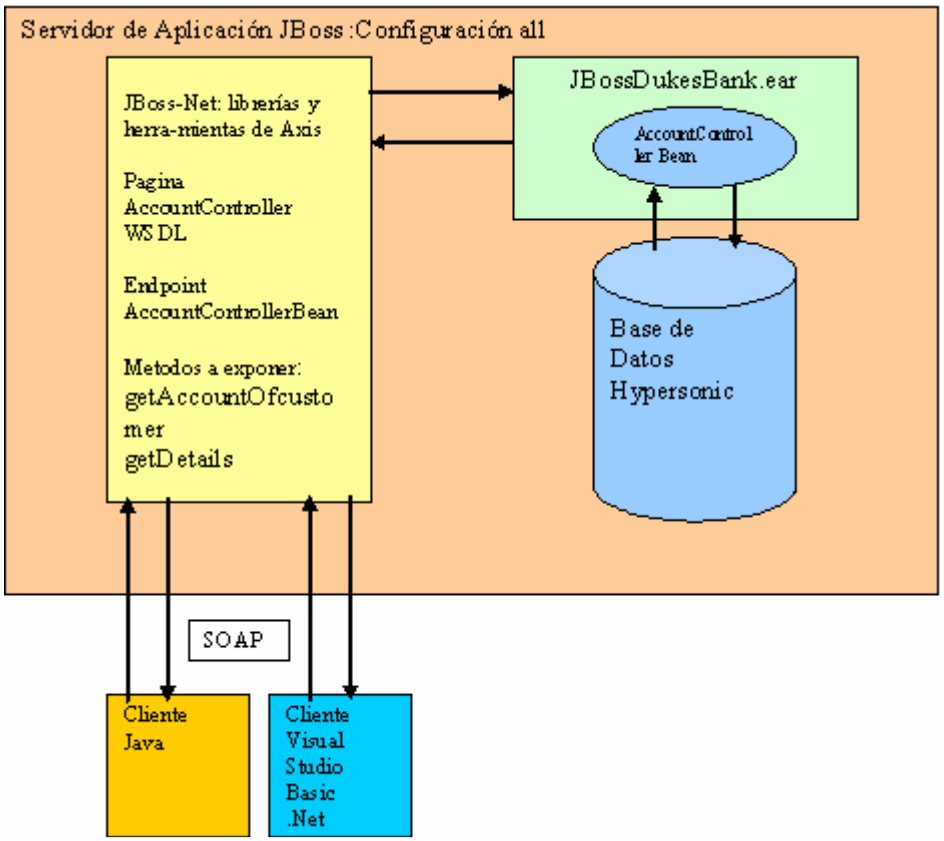
Aquí se ve la estructura de la base de datos de sus tablas y sus campos correspondientes así como la oportunidad de utilizar comandos SQL, create, drop, select, updatre etc.



Despliegue de la Aplicación en JBOSS

[3] **Anexo 7. Instalación de la aplicación Duke´s Bank en el servidor de aplicación JBoss**

El esquema a continuación muestra un descripción de cómo esta relacionado el software a utilizar y como intervieve para su desarrollo.



Descripción del WebService

La aplicación esta en marcha es ahora de llevar el concepto de Web Service a la práctica y esto es seleccionar el componente a exponerse, bien podría ser un session bean el cual esta a cargo de AccountController , el cual a través de sus método **getDetails** toma el número de la cuenta y la respuesta es un conjunto de datos: tipo, descripción, balance etc.

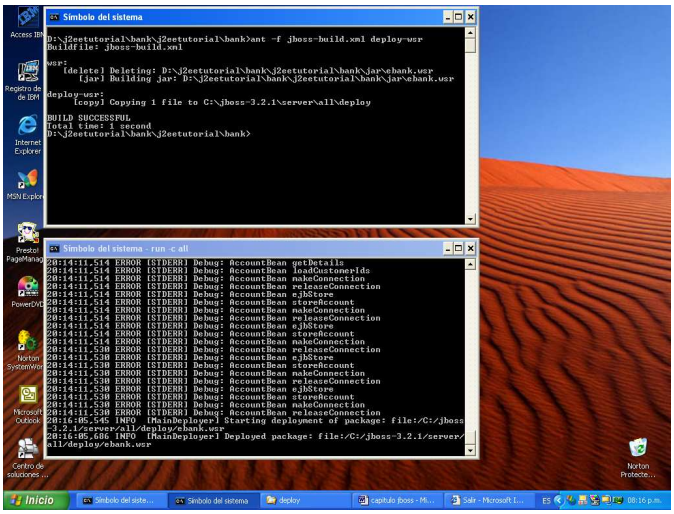
El ejemplo tiene a la aplicación desplegada en el servidor JBoss, el cual viene integrado con el paquete JBoss-Net. Este paquete reconoce el Web Service a través de un archivo con extensión WSR (Web Service Archive), así mismo, es simplemente un empaquetamiento con descriptores de despliegue y clases que describen el servicio y que reconoce inmediatamente el servidor.

La extensión wsr indica que es solo un archivo Jar estándar con la extensión wsr y un archivo META-INF/web-service.xml. Este último es un archivo de descripción que utiliza Axis (también llamado Web Service Deployment Desciptor ó WSDD)

El archivo **ebank.wsr** es copiado al directorio de despliegue de JBoss .este archivo es un jar pero JBoss-net necesita esta extensión para reconocer que en el viene integrado los descriptores de despliegue y los demás archivos de configuración escritos en lenguaje xml así como la página wsdl.

Es en el WSDD donde se aprecia en cada etiqueta el nombre de el servicio, que clases se requieren y donde localizarlas, en este caso, ubicar el EJB a través de JNDI (Java Name Directory Interfase), así como los métodos en este caso **getAccountsOfCustomer** y **getDetails**.

La simplicidad del motor de **Axis** hace que con solo este archivo y la herramienta **AdminClient** hacen posible crear una página WSDL

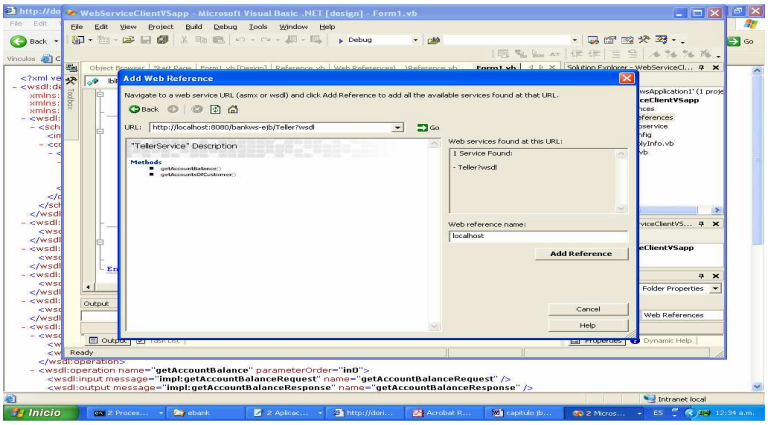


Despliegue del WebService Controller archivo ebank,wsr

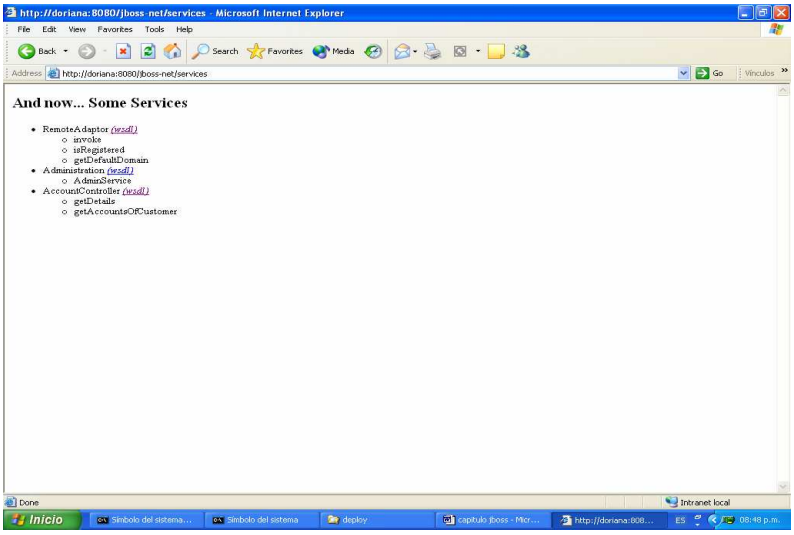
Para verificar los Web Services disponibles verificar la siguiente url: http://localhost:8080/jboss-net/services:

De esta forma apreciar también los métodos que van a interactuar con los clientes y que son parte de los componentes de la aplicación

Nuevamente comprobar el Web Service a través de la plataforma de VisualStudio donde el módulo Webreferences muestra otro Web Service localizado en la url mencionada, como lo explica anteriormente de la aplicación Dukes Bank.
Una forma de invocarlo es realizando el cliente Web Service a través de un explorador en Visual Basic Studio .Net , ést e mismo localiza todos los WebServices disponibles en la máquina local o bien via remota si fuese el caso..

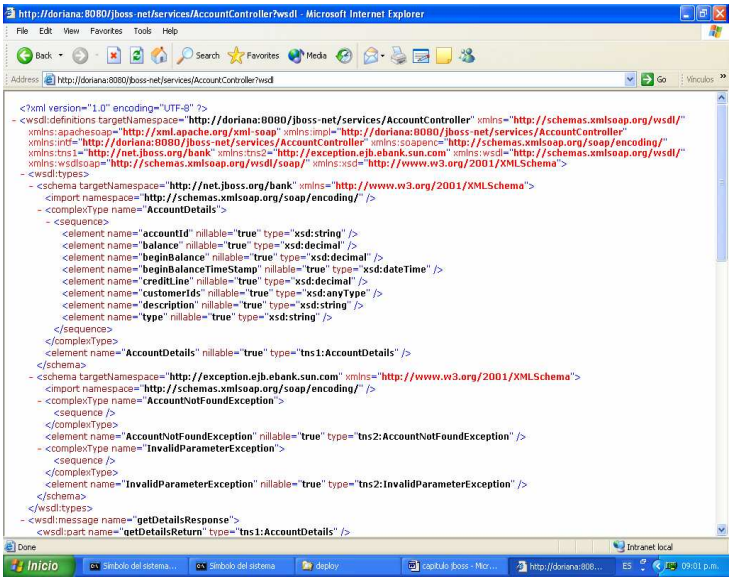


Localización del Web Service a través de Visual Basic Studio .Net



Web Services disponibles en JBOSS-NET

La página WSDL donde es un archivo xml define el protocolo SOAP, métodos que se exponen de la aplicación, la cantidad y el tipo de variables a utilizar tanto en la petición como en la respuesta, es decir, la información se requiere para desarrollar un cliente capaz de acceder a los métodos correspondientes, como se ha de apreciar la mayoria es información técnica. Es aquí donde la página actua como la interfase principal para la interoperabilidad de las plataformas con sus respectivas aplicación y con ello tenemos la capacidad de verificar los mensajes a utilizar, o bien, veificar que tipos de mensajes o datos se van a manipular y de que forma, el endpoint del WebService de la aplicación, donde se procesará la recepción y respuesta de datos que que en éste caso estaá a cargo de Account Controller.

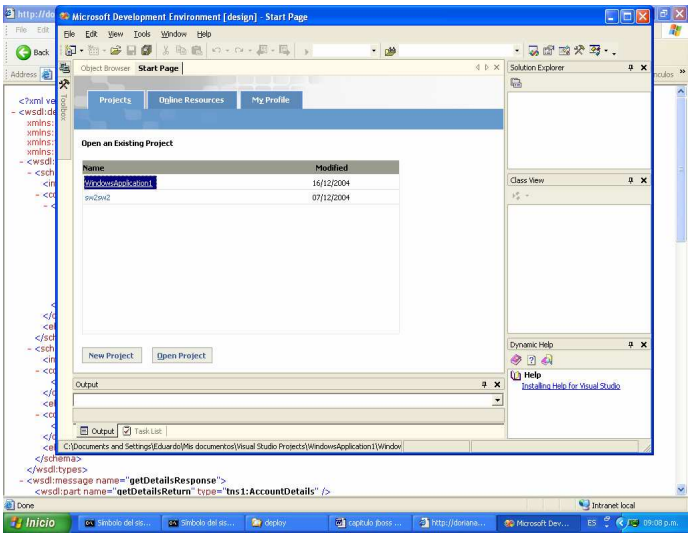


Página Controller.wsdl (describe el WebService)

Cliente Visual Basic Studio .Net

Las alternativas para probar el funcionamiento del Web Service es por medio de un desarrollo de programas clientes como se ha mencionado con anterioridad, dichos clientes han de ser utilizando diferentes lenguajes de programación e implementados en plataformas distintas como pude ser en Unix, Windows, OS, etc.

Para ello un buen ejemplo ha de establecer un cliente Visual Basic Studio .Net, en este ambiente de programación esta enfocado principalmente al uso de WebServices, donde trae implícito el conjunto de librerías SOAP, así como archivos xml capaces de desarrollar un WebService, crear un programa cliente el cual puede buscar y reconocer WebServices de forma local o remota.



Visual Basic.Net Inicio del proyecto cliente

En esta parte de VisualBasic.Net se encuentra una sección llamada WebReferences la cual se encarga de localizar los WebServices disponibles ya sea de forma local o bien de forma remota

Ingresamos la dirección donde esta los Web Services disponibles http://localhost:8080/jboss-net/services/AccountController?wsdl

El Web Service presenta los métodos en los cuales se procede ha realizar la programación correspondiente para invocar los datos de la cuenta. El los métodos de DukesBank son los siguientes:

getDetails( ) y getAccountsOf Customer( )

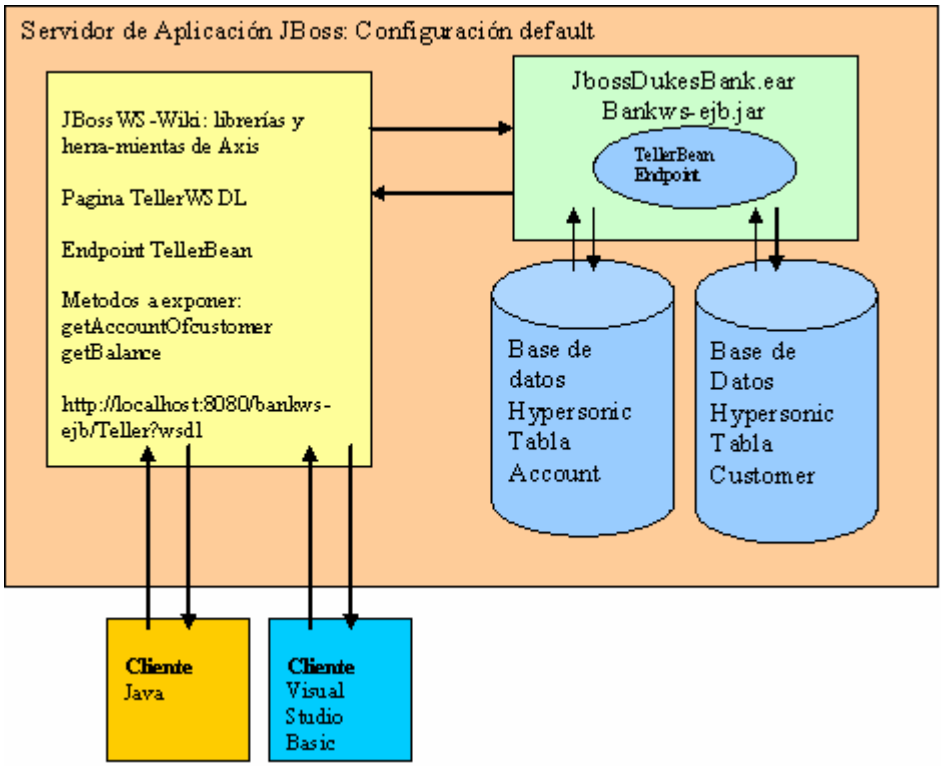
Por medio de VisualBasic Studio .Net realizar la consulta correspondiente: ingresar el número de cuenta, por ejemplo, la 200 que existe en la base de datos; es enviada como un objeto y transformada a su vez en mensaje SOAP , las líbrerías de SOAP de Visual Basic Studio anunciadas en el programa cliente actuan para establecer el mapeo respectivo . Teniendo una cadena que es un dato manipulado como cadena se recibe através del motor de Axis para realizar el segundo mapeo de cadena a un objeto java. De esta forma la aplicación tendrá una manipulación de la petición y hará uso de el EJB y demás componentes tanto el método getDetails que es el principal el regreso es un conjunto de objetos de tipo **AccountDetails** que requieren ser mapeado de un mensaje XML a objetos manipulables para VisualBasic.Net.

Como es de esperarse la modularidad de JBoss y el soporte de herramientas de software libre han permitido iniciar el proyecto de colocar la tecnología de Web Services al alcance, siempre y cuando revisando el soporte que han dado otras compañías a este proyecto.

3.6 EJEMPLO DE WEBSERVICE UTILIZANDO JBOSS WS TELLERSERVICE

JBossWS integrado en la versión 4.0.1 de JBoss permite tener la oportunidad de probar el comportamiento de la implementación de la aplicación Dukes Bank ahora utilizando otro Bean llamado **Teller Bean**.

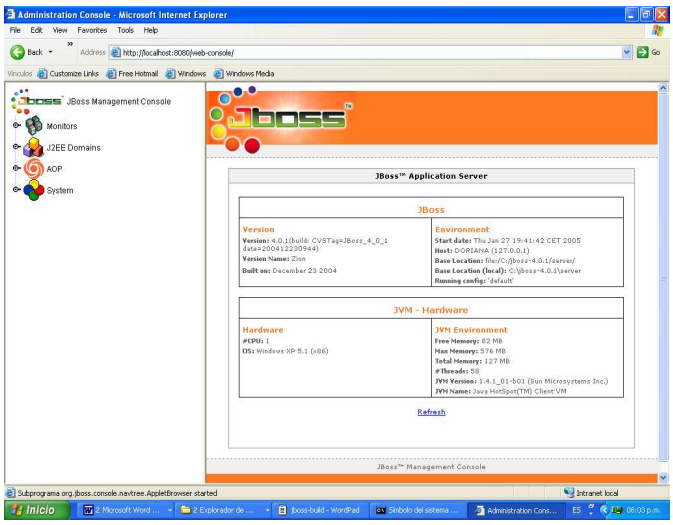
Aquí se aprecia la presentación de la consola con información acerca la versión, el entorno, el tipo de maquina java utilizado, el hardware del sistema, así como los servicios disponibles como es el de JBossWS. Para esta prueba se han modificao el nombre del archivo .ear , ahora es **JBossDukesBank.ear** donde contiene la aplicación.



Grafica de la implementación Web Service Teller en JBoss

La idea es mostrar caracteristicas distintas en estapa versión, las mejoras que posee es que tiene mayores caracteríticas orientadssa la arquitectura J2EE. Esto servirá para ver la evolución que presenta este servido con su módulo par WebServices JBOSS-WS Wiki que está basado de igual forma en Axis. La implementación de la aplicación en este servidor y su puesta en marcha hace uso de los mismos procedimientos anteriormente mencionados, teniendo ciertas variaciones[4]

[4] **ver Anexo 8. Implementacion del Session Bean Teller como WebService en JBoss**



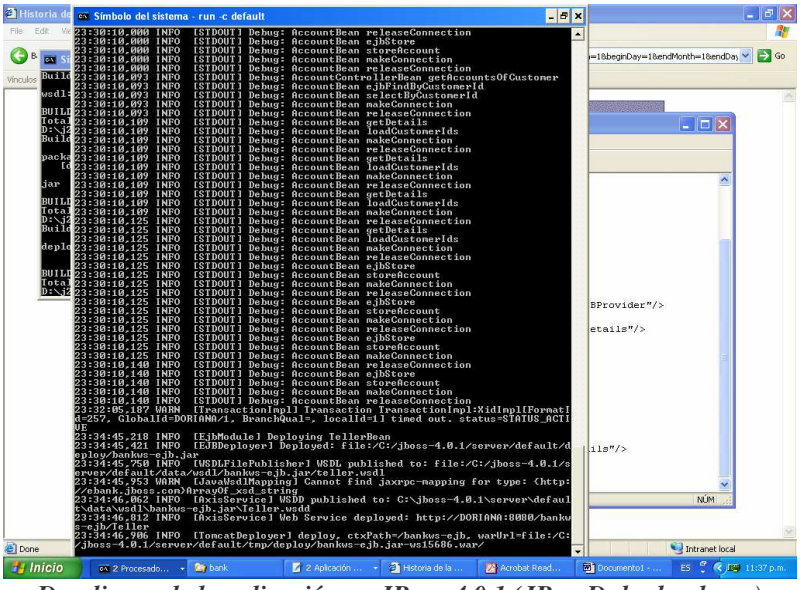
Consola Web de JBoss-4.0.1

Ahora, desplegar en este caso un Session Bean llamado TellerBean de tipo stateless, este mismo se encuentra ligado a la aplicación pero su desarrollo se ha manejado de forma externa, es decir, tantos sus descriptores de despliegue se encuentra en otra capa y por ende el archivo de despliegue bankws-ejb.jar donde el Web Service se encuentra ya empaquetado listo para desplegarse en la carpeta deploy del servidor. Este Bean tiene la misma funcionalidad que **AccountController**.

Cabe mencionar que la configuración de JBOSS en default ya que esta posee los módulos JBossWS y no hay necesidad de correr todos los servicios como en la configuración all.

TellerBean está incluido en una carpeta externa la cual es una forma de manipular solo la parte a exponer de los servicios de una aplicación. Es una alternativa en la cual puede manejarse el EJB pero ahora llamado TellerBean y crear la clase endpoint.

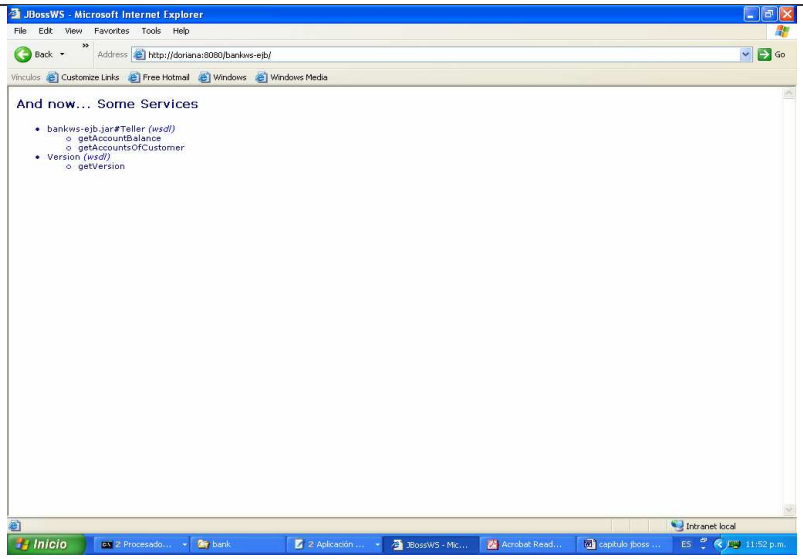
Al momento de copiar los archivos .ear y .jar en la carpeta deploy hacen ver el reconocimiento de los beans así como de los componentes. Verificar si arroja el servidor un mensaje de que los archivos han sido correctamente desplegados



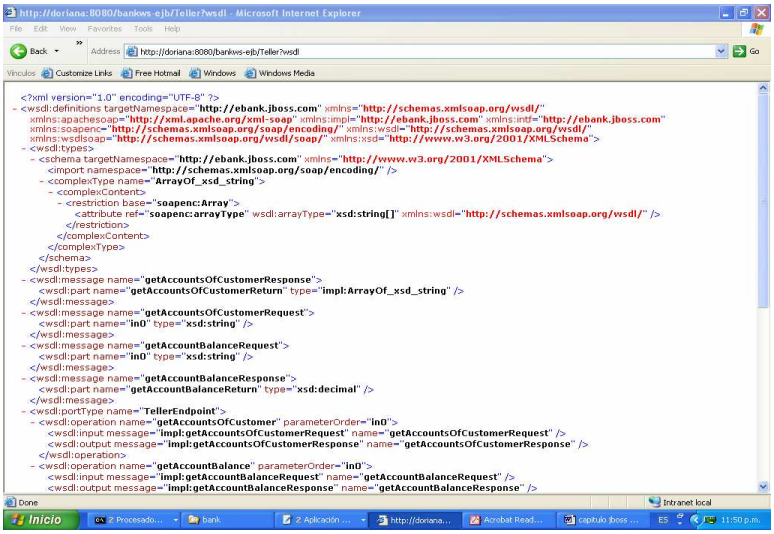
Despliegue de la aplicación en JBoss-4.0.1 (JBossDukesbank.ear)

Para acceder al servicio escribir la siguiente URL: http://doriana:8080/bankws-ejb/Teller?wsdl la cual proporciona una vez más la interfase de nuestro WebService corriendo en un servidor de software libre y con herramientas proporcionadas por Apache.

Pagina :http://doriana:8080/bankws-ejb/Teller?wsdl
And now... Some Services bankws-ejb.jar#Teller *(wsdl)* getAccountBalance getAccountsOfCustomer Version *(wsdl)* GetVersion

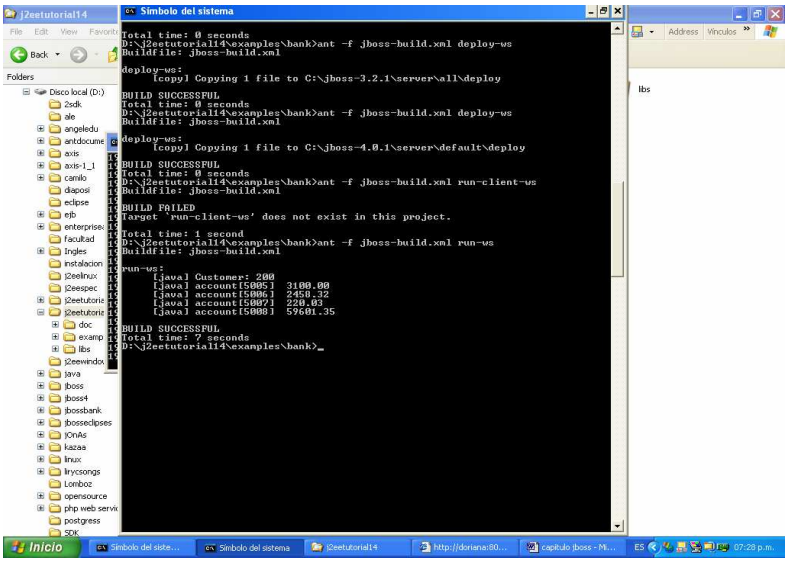


Despliegue de los WebServices Disponibles por el servidor vista en el explorador

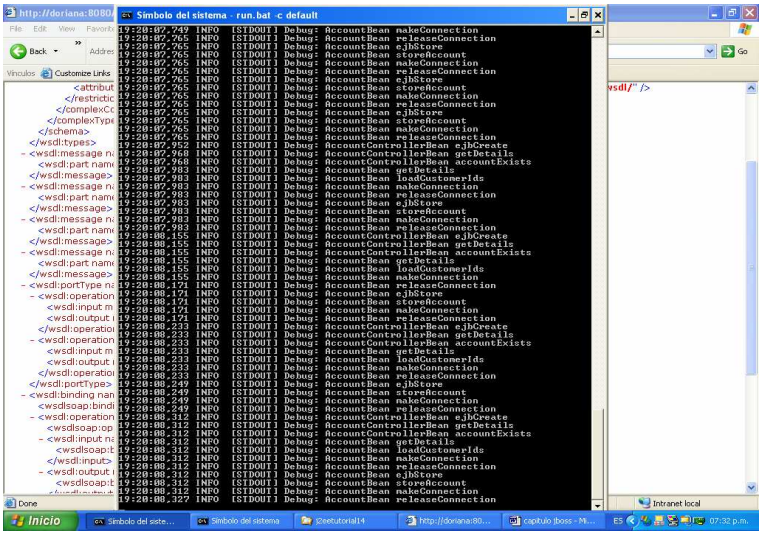


Página WSDL :Archivo Teller.wsdl

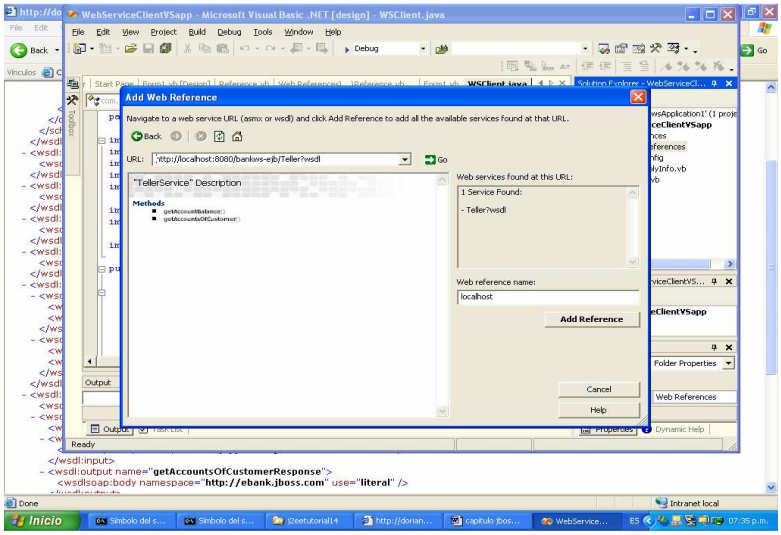
Teniendo el WebService llamado Teller iniciar el proceso de la creación de un cliente java o bien un cliente web. Para ello es primordial establecer un cliente java para comprobar la comunicación que existe dentro de un cliente escrito en el mismo lenguaje de programación y corriendo localmente.



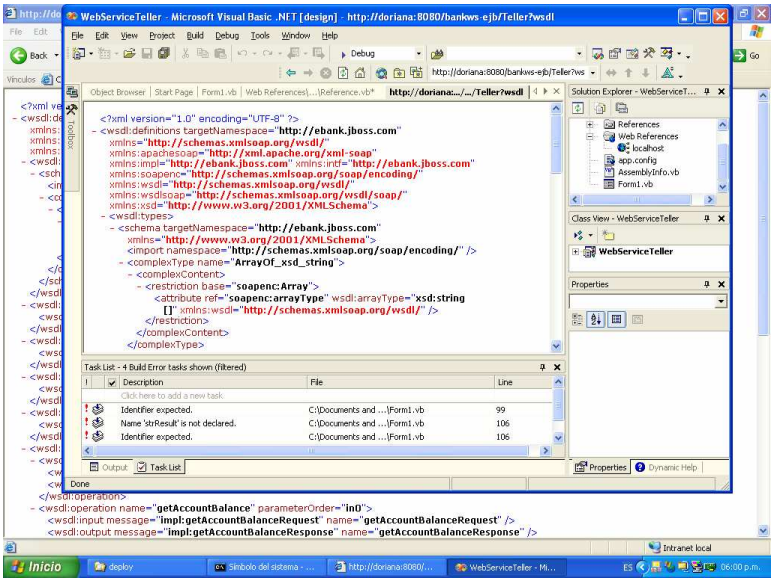
Corrida programa cliente java del WebService Teller



Comportamiento del servidor JBoss-4.0.1

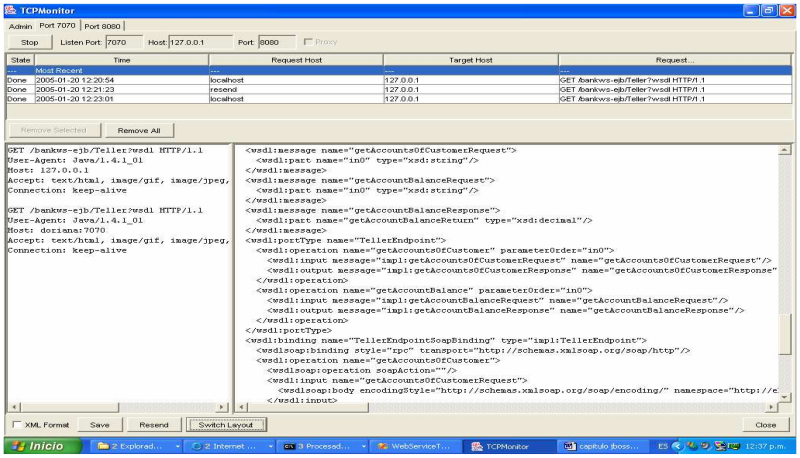


Localización del servicio a través Visual Studio



Navegador en Visual Studio desplegando la página Teller.wsdl

Para ver el proceso en que ocurre en el envió de mensajes SOAP y por ende la interoperabilidad de los sistemas con un programa cliente, se puede hacer el uso de una herramienta mas por parte de Axis cuyo nombre es Tcpmonitor lo que permite apreciar la Comunicación y los mensajes recibido y enviados a través de SOAP, la figura muestra lo anterior



TPCMonitor: Herramienta de Axis permite exponer el comportamiento de los mensajes SOAP que componen al WebService Teller.

IV ANÁLISIS DE FACTIBILIDAD DE APLICACIÓN DE UN WEBSERVICE

Los Web Services resultan ser una tecnología atractiva como una alternativa de solución mas adecuada para la integración de aplicaciones. Más allá de una integración de aplicaciones, es hablar sobre las posibles aplicaciones y los beneficios orientados a los negocios electrónicos, y a mejorar procesos en la empresa utilizando un esquema como éste donde en la comunicación de plataformas interviene protocolos capaces de brindar el medio para compartir, difundir y recibir información desde cualquier otro sistema o plataforma.

Antes de ello, es importante establecer que factores interviene en un desarrollo de este tipo: dentro del ámbito empresarial y de negocio.

Necesidades de las empresas

Extender aplicaciones existentes en la Red, en Internet, Intranet o extranet de la empresa

Integrar las aplicaciones existentes

Crear nuevas aplicaciones en la Red

Integración multicanal (móviles y PDA´s)

Ahorrar costes y aumentar productividad

Soluciones especificas de la empresa

Soluciones de Intranet y portales de empleados

Portales de clientes y proveedores

Soluciones de marketing y relacional multicanal

Creación de Webs empresariales y gestión de contenidos

4.1 PLANEACIÓN ESTRATEGICA DE UN WEBSERVICE

La intervención del Internet en el ambiente, ha modificado la manera tradicional de hacer negocios, y al mismo tiempo la forma en que la información debe moverse. Por tanto, las empresas se enfrentan con la necesidad de desarrollar nuevas estrategias de servicios para poder destacar, y no sólo cubrir con las necesidades requeridas por sus clientes, sino también por las necesidades que surgen dentro de la empresa misma.

Sin embargo, las empresas se han dado cuenta que la implementación de los Web Services no es una tarea fácil, se han encontrado con numerosos obstáculos causados por una falta de conocimiento del impacto que éstos ocasionarán a lo largo del negocio. Surge entonces una nueva necesidad, la de planear estratégicamente la incursión de un Web Services dentro del negocio.

Hoy en día, la planeación estratégica ha sido una herramienta que ha evolucionado, y comprende desde la planeación de procesos y análisis situacional para incursionar en nuevos mercados hasta la implementación de Tecnologías de Información (TI) y Web Services.
Dentro de una Planeación Estratégica, se debe determinar que debe hacerse, cómo se va a hacer y cómo puede darse cuenta una empresa, si el Web Service cumple o no con los objetivos deseados y establecidos.

Donde una genuina estrategia, implica con el involucramiento de todos los procesos del negocio, así mismo el reajuste de estos (reingeniería). Este proceso comienza con un análisis de la situación actual de la empresa, evaluando el entendimiento de la estrategia, los procesos operativos y la aceptación del Web Service dentro la organización y la aplicación del sistema, continuando con las modificaciones necesarias para adaptarse al Web Service, y la infraestructura de TI requerida, esto con el fin de mejorar la eficiencia y la productividad de

la empresa. Y es así como la empresas comienza a conceptualizar la forma de mejorar sus servicios que le otorgarán una diferenciación sobre sus competidores.

Una vez realizado este proceso, se diseña la arquitectura de información que identifica las necesidades globales de información de la empresa, desarrollándose un modelo de TI que tiene como objetivo la transformación de las estrategias de negocios en una estrategia tecnológica. Posteriormente se construye la arquitectura, se definen los elementos claves, las características esenciales de la misma (hardware y software, etc.), y se continúa con el diseño a detalle de los modelos de operación del Web Service, que describen el funcionamiento del área informática. Por último se elabora un modelo de planeación, donde se establecen las prioridades para la implementación de la TI para el Web Service y los procesos operativos. Se define un plan de acción tomando en cuenta los riesgos y el orden de desarrollo de los proyectos, y el retorno de inversión de dicha implementación.

Dentro de un modelo de planeación estratégica de TI se visualiza una administración de las mismas, donde estas se planean inicialmente para lograr una ventaja competitiva. Esto se lleva a cabo estableciendo la estrategia de negocios deseada, enfocándose a la revisión actual de la empresa y estableciendo el punto o meta a dónde se quiera llegar y los elementos que intervendrán para lograr su objetivo.

Modelo de Planeación Estratégica para Web Service.

Al mezclar "e-technologies" con conceptos tradicionales de negocio, se afecta la estructura tradicional de negocios. En respuesta a este impacto, han surgido innumerables modelos de planeación estratégica de "e-technologies", que facilitan su implementación. Los modelos de planeación estratégica han permitido a las organizaciones alinear sus estrategias de negocio a un ambiente competitivo, y les ha permitido desarrollar una ventaja competitiva.
El Modelo de Planeación Estratégica de Web Service tiene como objetivo guiar a la empresa a través de conjunto de pasos ordenados que lo componen, culminando en el desarrollo y ejecución de un Plan de Negocio para la implementación de un Web Service. El modelo está provisto de 4 fases partiendo de que el Web Service" proviene de una idea o concepto electrónico CE a implementar. La siguiente tabla explica los elementos involucrados en este modelo

FASE 1 **Formulación Estratégica del concepto electrónico (CE)**	**FASE 2** **Ambiente Competitivo**	**FASE 3** **Implicaciones del CE en Alineación al Negocio.**	**FASE 4.** **Factibilidad de implementación**
• Descripción actual del negocio • Procesos claves del negocio • Analizar	• Comportamiento del clientes • Ambiente competitivo industrial	• Forma en que el CE impactará positivamente al negocio • Factores que implementará el CE de acuerdo al impacto.	• Especificación de la infraestructura del CE (Software , Hardaware) • Equipo de trabajo necesario • Aspectos legales • Planeación Financiera • Plan de negocios.

Planteamiento de un Modelo de Planeación Estratégica de un "Web Services" en la empresa.

Formulación estratégica del Concepto Electrónico (CE)

La formulación estratégica consiste en el reconocimiento de declaraciones estratégicas para el negocio, objetivos específicos a ser alcanzados y los movimientos necesarios para entender el futuro y las etapas en que cada objetivo se moverá. La formulación de la estrategia se plantea para enmarcar todos los factores dominantes de una empresa con una implicación secuencial en la misma, tanto de negocio como de perspectivas funcionales. El objetivo de esta fase, es definir la estrategia basada en un CE, y el propósito de esta dentro del negocio.

Ambiente Competitivo.

Hoy en día el proceso de las decisiones de compra está básicamente basado en estímulos con respecto a la reacción del cliente y a su vez por las características del mismo, el ambiente que lo rodea, la tecnología, y la logística que existe para que él pueda realizar su compra. Por otra parte, es importante también para las empresas, no sólo evaluar el comportamiento de sus clientes, sino también el ambiente competitivo del cual se rodea, es decir conocer a sus competidores para tener éxito en su estrategia basada en TI.

Esta fase analiza 2 puntos basándose en otros modelos de apoyo:

Comportamiento del cliente actualmente el consumidor o cliente representa un punto clave en el negocio. El comportamiento del cliente se derivará el comportamiento futuro del vendedor, es decir la empresa proveedora del servicio. Este análisis hace un enfoque del comportamiento del consumidor visto electrónicamente.

Implicaciones del Concepto Electronico en alineación al negocio.

Existen muchas implicaciones para la implementación de un Web Service. Ciertamente, la importancia de las TI que envuelven al Web Service son relevantes para crear un ambiente sólido. Sin embargo, existen otras implicaciones adicionales a estas que son trascendentes dentro de la empresa. El análisis y detección de estas implicaciones es de vital importancia para lograr una alineación estratégica entre la TI, el Web Service con los procesos del negocio.

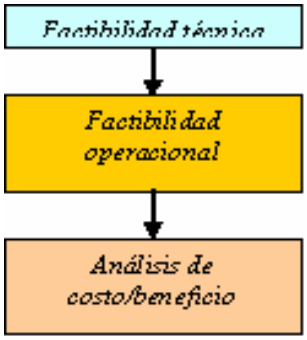
Las implicaciones más importantes a considerar en el análisis del modelo propuesto de acuerdo a la implementación alineada al negocio son:

El CE impactará positivamente al negocio. Cualquier empresa posee fortalezas, oportunidades, debilidades y amenazas en su ambiente, y es común estos factores, y es importante el análisis de estos factores. El objetivo de este punto es detectar las oportunidades que se tendrán al implementar un Web Service determinado y cómo este se alineará con el negocio. Muchas ideas inician con un propósito positivo, es decir una oportunidad. Lo importante de este propósito es la forma en que esta idea modificará e impactará a la empresa que la implemente. Este punto detecta el "cómo" y el "dónde" impactará un Web Service, y los riesgos que existirán dentro del negocio que lo implemente., esto haciendo pensar en un desarrollo a largo plazo un beneficio que repercute en soporte, productividad y una larga vida del proyecto, que también pueda ser renovable, escalable y mantenga sus objetivos principales de la empresa.

Al analizar el punto anterior, se logra identificar diferentes escenarios en los que se puede mover la empresa a su conveniencia y necesidad. El objetivo de este punto es analizar qué elementos de TI se verán involucrados para lograr estos escenarios.

Factibilidad de implementación

La factibilidad de implementación es un punto que involucra finanzas, y la factibilidad de realizar o no un proyecto determinado. Una empresa que adopta un concepto electrónico como estrategia de ventaja competitiva, debe saber a detalle qué tan económicamente posible es su realización o no. De acuerdo con Laudon (2001) y O'Brien (2002) [5,6], las prioridades de factibilidad en una organización están dadas por:



Prioridades de factibilidad en la organización

El modelo propuesto añade a estas prioridades factores complementarios para un análisis más completo en la factibilidad de implementación de un Web Services. Tales factores son:

Especificación de la infraestructura del Concepto electrónico: Este punto hace referencia a la factibilidad técnica que rodea al concepto electrónico. Especificaciones de hardware y software son analizados en este punto, listando todos los requerimientos necesarios para cubrir lo descrito en la fase 3.

Equipo de trabajo necesario: La implementación siempre requiere de un grupo de personas que hagan posible la realización de un proyecto. Este grupo involucra tanto a la empresa misma como a un grupo consultor y administradores de información. Algunas ocasiones, las empresas optan por especialista en algunas áreas, y realizan outsourcing.

Aspectos legales: Internet ha causado modificaciones dentro del marco legal. Cuando los compradores y vendedores no se ven entre sí, pueden existir acciones ilícitas, tales como el fraude, robo, etc., a través de Internet. Algunos países han logrado establecer reglamentos y políticas de uso, acción e implementación de servicios y tecnologías en Internet. Es indispensable para cualquier empresa que desee implementar un Web Services conocer las regulaciones y el estatus legal que involucra tal implementación.

Planeación Financiera: El análisis de factibilidad viene acompañado de un plan de financiamiento para dicha implementación, tomando en cuenta los 3 puntos anteriores. La planeación financiera establece el costo de la inversión en su totalidad y el retorno de la inversión.

Plan de Negocios: Cuando la empresa determina que es factible de realizar el proyecto de implementación de un e-servicio, el resultado final de una planeación estratégica es un plan de negocios. En el plan de negocios se mencionan todos los puntos a seguir para hacer posible la implementación del Web Service.

Así también es necesario descubrir otras alternativas a esta tecnología como por ejemplo tener el uso de un esquema simple cliente-servidor o desarrollo en tres capas conocido como front-end , middlware y back-end , es decir, las empresas tienen que descubrir sus alcances, sus situación como negocios si es factible agregar e implementar un esquema Web Service para agilizar sus procesos dentro de su organización y del crecimiento del negocio a través de mejores propuestas de interactuar con clientes y proveedores a sí como la productividad de sus áreas y empleados.

Palabras Claves: Planeación estratégica, tecnologías de información.

Los Web Service utilizan el lenguaje XML en el cual se ha establecido un protocolo de comunicación SOAP y la ventaja de obtener estándares. La idea de las empresas involucradas en este proyecto es poder obtener un beneficio a los negocios electrónicos comúnmente llamados e-bussiness, es utilizar una innovación o bien una opción que facilite e intervenga en promover servicios , productos de varias compañías para hacerse de alguna publicidad imitando el proceso de una sección amarilla donde será posible dentro de un registro o repositorio incluir diversas páginas WSDL capaces de mostrar le giro del negocio, cual es su oferta y determinar que funciones tiene.

- Descubrir la problemática en la cual se sitúa la compañía, y en que contexto puede aplicarse un Web Service en el sistema, el principalmente de ellos es incrementar de alguna forma su negocio, mediante captación de clientes y proveedores.

- Determinar con que tipo de aplicación o sistema cuenta el negocio o la compañía con la finalidad de exponer módulos, es decir, la empresa debe evaluar su objetivo, si uno de sus objetivos esta involucrado, establecer una comunicación mas directa y efectiva con otros sistemas para lograr mejorar procesos tanto de promoción, solicitar y proveer tanto productos como servicios, información de precios, que requieren evaluarse directamente en los sistemas.

- Establecer la infraestructura con que se cuenta, software, Hardware y si la plataforma es propicia para llevar a cabo un proyecto de esta dimensión y propósito.

- Realizar un estudio Costos-Beneficio que interviene en adquirir licencias y procesos de instalación,. Desarrollo, para la exposición de módulos en Web Service y los beneficios a corto y mediano plazo, una opción estudiada dentro del proyecto es establecer una alternativa de usar software y herramientas con licencia libre.
- Evaluar comparativas que se ajusten a los alcances de crecimiento, inversión en el uso de esta tecnología.

Varios proveedores líderes promocionan los Web Services como un modelo de programación para la comunicación entre aplicaciones. Estas compañías piensan que la conexión de aplicaciones a través de la Internet mejorará la capacidad de las empresas para trabajar conjuntamente con sus socios de negocio, proveedores y clientes. Creando una capa de Web Services sobre una aplicación corporativa existente, las organizaciones podrán permitir que sistemas externos puedan invocar las funciones de la aplicación a través de Internet (o una intranet corporativa) sin tener que modificar la aplicación misma. Por ejemplo, varias compañías están hoy en día creando Web Services que actúan como front end para aplicaciones de entrada de órdenes que están residentes internamente en un mainframe. Estas compañías permiten a los sistemas de compras de sus clientes enviar órdenes de compra a través de la Internet. Poner una capa de Web Services sobre las aplicaciones existentes es una solución muy interesante para integrar las aplicaciones desarrolladas por los diferentes departamentos y así reducir los costos de integración.

4.2 METAS Y RETOS DE LOS WEB SERVICES

METAS EN LOS NEGOCIOS DE ALTO NIVEL
• Reducir costo de Integración
• Incrementar la flexibilidad de las empresas para capitalizar las oportunidades de negocio.
• Reducir el tiempo para mercadear nuevos productos y servicios.
• Incrementar el ROI (Retorno sobre la inversión) para los actuales y futuros inversionistas

RETOS DEL ENTORNO ACTUAL
• Incrementar el foco de mercado en desempeño financiero de las empresas
• Incrementar en la complejidad del portafolio de aplicaciones existentes sin pagar a propietarios de estándares de programación o de interfases.
• Incrementar la integración de la información dentro de un entorno de negocio extremadamente dinámico y competitivo
• Grandes inversiones en la arquitectura de tecnología actual TI

RETOS TECNICOS
Para los webServices tengan éxito, se requiere vencer algunos retos técnicos que aun las compañías líderes están soportando dicha aplicación, entre las cuales podemos encontrar:

Descubrimiento.
Establecer estrategias si un servicio se ha modificado o se cambia después de haber sido anunciado. Existe dos estándares como es WSDL y UDDI que es aquí donde se sigue trabajando por parte de varias organizaciones en este aspecto.

Confiabilidad

Algunos sistemas huésped de WebServices son y serían más confiables que otros , esto da la capacidad tanto del hardware y software utilizado y la forma en que trabaje dicha aplicación. hay que destacar un grado de confiabilidad y como puede ser comunicada, así también puede ocurrir un caso en que las compañías prevean cierta confiabilidad , dependerá de ellas dar esa característica. Además de informar si existen otros WebService alternos hospedados en otra compañía.

Seguridad

Algunos WebServices estarán públicamente disponibles y con poca seguridad, hay que destacar que los webServices desarrollados con software comercial utilizarán comunicaciones encriptadas con autenticación. Es hoy en día el uso continuo de http sobre SSL que proveerá la seguridad básica, tanto de la aplicación como los usuarios. Así también, mencionar, HTTPS, credenciales de cliente y certificación son un a mejora a la solución de seguridad. Sin embargo, si la seguridad es a nivel aplicación, podrán utilizarse funciones, encabezados SOAP o autenticación personalizada. Así que las herramientas de seguridad de aplicaciones pueden aplicarse de cierta forma adaptando al WebServices del sistema en particular.

Transacciones

Los sistemas tradicionales de transacciones utilizan un método de compromiso de dos fases, se recolectan todos los recursos participantes y se aseguran estos hasta que se lleva acabo la transacción completa. Terminada la transacción se liberan los recursos. Este método puede funcionar bien en ambientes cerrados donde las transacciones son de corta duración, pero no trabaja bien en ambientes abiertos donde las transacciones pueden tomar horas, si no es que días.

Administración

Es importante resaltar en si es posible que otros puedan administrar webServices ajenos a las compañías, y pensar en mecanismos adecuados para la administración de ambientes sumamente distribuidos.

Contabilidad

Definir el periodo de tiempo en que un usuario puede acceder y ejecuta un Webservices . como corra un webServices y que parámetros se ha de considerar ya sea por suscripciones o pagos por eventos.

Pruebas.
Depurar un WebServices que proviene de diferentes compañías y hospedado en diferentes ambientes y en diversos sistemas operativos.

La forma de superar estos retos es encontrar una estrategia adecuada, analizando circunstancias y situaciones donde se aplica un WebServices, conocer ampliamente la aplicación para una eficaz implementación y el resultado es buscar:sistemas

- Tolerante a fallas

- Paralelismo masivo
- Distribuidos
- Bien organizados
- Auto-reparables
- Diseñado en capas
- Diseñado a partir de componentes simples

El problema de integración de aplicaciones y sistemas distribuidos

Las interfases de las aplicaciones no son limpias

- Desde que todo es integrado escribiendo código las interfases de los módulos puede definirse incorrectamente o documentarse mal

El desarrollo es una labor extremadamente intensa y extensa

- El desarrollo e implementación de aplicaciones requiere de instalación de software, codificación manual y posibles cambios a las aplicaciones existentes.

Un gran número de aplicaciones heredadas aun existen al interior de las organizaciones.

- Las empresas han invertido en un gran número de aplicaciones heredadas que requieren un gran trabajo para la integración con aplicaciones nuevas para poder utilizar su capacidad en los negocios.

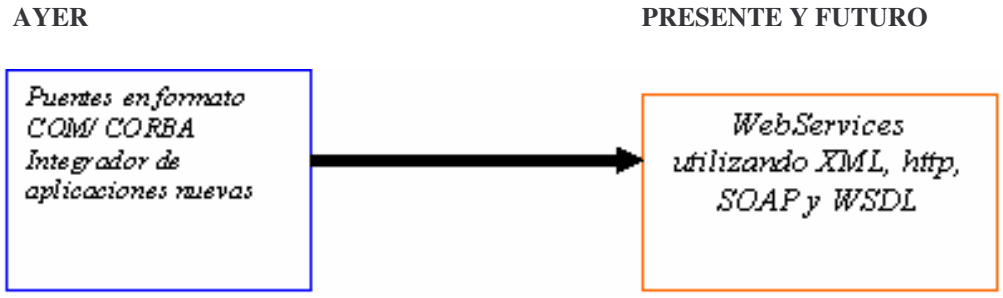
Las empresas y sus aliados de negocios han adoptado múltiples estándares de programación.

- Los distintos lenguajes y estándares entre los ambientes de la organización hacen extremadamente difícil lograr la interoperatividad entre las aplicaciones y servicios, tanto en el interior de la empresa como los aliados del negocio.

Aplicaciones de Web Services

Simplemente un Web Service hace la conexiones para negocios utilizando estándares como SOAP, WSDL y UDDI.

Integración de aplicaciones nuevas esta gravitando alrededor de los Web Services.



Tendencia de tecnología orientada a los sistemas distribuidos

UNA COMPARACION DE TECNOLOGÍAS DE INTEGRACIÓN DE APLICACIÓN A WEBSERVICES	
CORBA	**Web Service**
Estandarizado por OMG (http://www.omg.org)	Provee éxito comercial
Éxito comercial en 1995	Protocolo de comunicación: SOAP
Existen múltiples implementaciones comerciales y licencia libre, disponibles para los lenguajes y sistemas operativos más usuales	Conceptualmente permite enviar peticiones/respuestas en XML (normalmente sobre HTTP)
CORBA ha sido y continúa siendo una buena tecnología para abordar integraciones complejas en intranets	Existen APIs, para los lenguajes más usuales
Sin embargo, no ha tenido éxito para integración de aplicaciones en Internet	Disponible para J2EE, .NET y LAMP(Linux Apache MySQL y Postgress)
Existen firewalls que no reconocen IIOP(que es el protocolo de comunicación)	Todos los fabricantes de tecnología proporcionan soporte para Web Service
Microsoft no fabrica implementaciones de CORBA	Las integraciones complejas en intranets suelen requerir funcionalidad que todavía no soportan los WebServices
Para abordar integraciones de aplicaciones en Internet es preciso usar una tecnología que cuente con el apoyo de todos los fabricantes de tecnología (Sun, Oracle, IBM, Microsoft, etc.)	

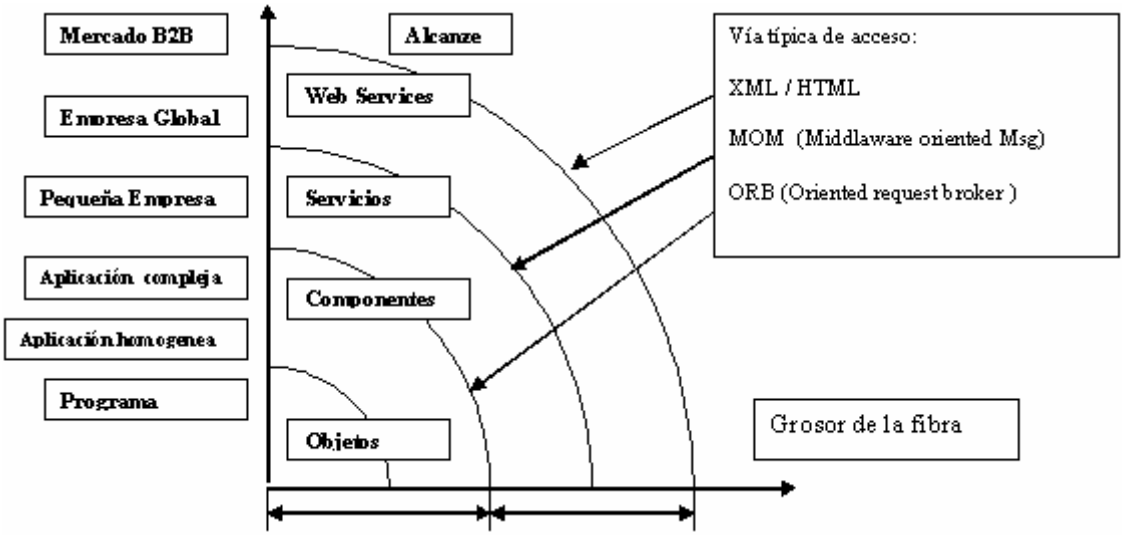
Tabla comparativa de tecnologías orientadas a los sistemas distribuidos

Adaptar las aplicaciones existentes para que funcionen como Web Services permitirá que los usuarios construyan nuevas y más poderosas aplicaciones que utilicen Web Services como una construcción en bloques. Por ejemplo un usuario puede desarrollar una aplicación de compras para obtener automáticamente la información sobre precios de varios vendedores, permitirá al usuario seleccionar el vendedor que quiera comprar, enviar una orden de compra y rastrear el envió mientras se realiza.

La aplicación de ventas también permitirá al vendedor, además de ofrecer sus servicios en Internet, revisar el crédito del cliente en línea, cargar el monto a la cuenta del cliente y coordinar el envió de la mercancía con una compañía de envíos.

Los Web Services permiten la integración de las empresas globales y sus proveedores y aliados de negocios utilizando un sistema de mensajes emparejados libres.

En la gráfica siguiente permite establecer el grado de alcance que puede presentar para las empresas globales la aplicación de los Web Services en cuestión de comunicación e interacción de negocios electrónicos y demás aplicaciones.



Gráfica de Relación de alcance de los WebServices

4.3 LA PROPUESTA DEL VALOR DE LOS WEB SERVICES

Abertura de puentes entre opciones de negocios

Los Web Service conectarán los datos de la cadena de producción directamente con los datos de ventas, para permitirle a las empresas manejar producción en tiempo real y así mismo tomar decisiones en esa misma frecuencia evitando pérdidas

Desbloqueo de datos para clientes mejor atendidos

Utilizando los Web Services para levantar una base de datos de perfiles de clientes y de un catalogo de productos, las empresas pueden generar precios bajos para contratistas, generando lealtad y haciendo ofertas proactivas basadas en proyectos específicos.

Hacer del outsourcing de aplicaciones

Utilizando una plataforma de Web Services, las empresas pueden crear notas investigativas personalizadas, para generar fidelidad al conectar la fuerza de ventas con el equipo de investigación interno.

4.4 OPORTUNIDADES PARA IMPLEMENTAR WEB SERVICE

PUNTOS A FAVOR **PUNTOS EN CONTRA**

Recurrente: es indicado y repetido a través de una gran base de usuarios

Las empresas pueden ofrecer herramientas Por el manejo de garantías basadas en Web Services para miles de usuarios y distribuidores a través de un mecanismo de Salida para registrar, rastrear e interponer quejas	Compartir diseños entre ingenieros de Compaq e Intel no tendría sentido por que la colaboración en este caso serán de persona a persona y única, y no masiva.

Dinámico: Información que cambia constantemente y de cuya actualización los usuarios se benefician constantemente

BM Amoco puede implementar un Web Service para brindar herramientas de manejo de riesgos como cambios para ayudar a los clientes a mitigar las fluctuaciones de los precios	DuPont publicando tablas de datos químicos que muy raramente cambian y tiene poco valor para los clientes

Desconectado: Hace un puente para una relación de negocios que no este conectado a Internet

La herramienta de rastreo de entregas de Ford Motor Company utiliza un Web Services para vincular los sistemas logísticos de UPS a la red expansiva de Ford de distribuidores individuales	Los Web Services no le ayudan a Ford a ahorrar dinero intercambiando con abastecedores como Delphi Automotive Systems , por que para este tipo de transacciones ya existe una red de intercambio de datos

Estos son ejemplos de algunas fuentes donde se considera algunas puntos donde las empresas deben evaluar sus necesidades para recurrir a los aspectos tecnológicos, en este caso, realizar un análisis en que situaciones pueden aplicar el uso de un Web Services de acuerdo a sus sistemas, así como también el objetivo principal de lo que se quiere establecer o si resuelve un problema del mismo negocio y funciones de sus procesos internos.

4.5 ALTERNATIVAS DE SOFWARE PARA EL PROYECTO WEB SERVICES FRENTE A J2EE Y SOFWARE LIBRE

Al apreciar el apoyo de compañías importantes como IBM, Microsoft, Sun Mircosystems , Oracle, donde sus servidores de aplicación presentan módulos adicionales orientados a la implementación de un Web Services donde incluyendo toda el API o librerías del protocolo SOAP , así como , una serie software diseñado para una fácil implementación para creación de clientes Web Services, con la finalidad de proveer a sus sistemas una innovación en las tecnología de información y enfrentar el problema de alguna forma de comunicación entre aplicaciones. Empresas líderes involucradas en el desarrollo de Web Services:

IBM

Sun Microsystems

Oracle

Microsoft

Servidores de aplicaciones certificados J2EE
BEA WebLogic Server.
Inprise/Borland AppServer.
IBM WebSphere ApplicationServer.
IONA iPortal Application Server.
Sun ONE Application Server.
Macromedia JRun Server.
Oracle Application Server.
Sun Java 2 SDK Enterprise Edition.

4.5.1 IMPLEMENTACIÓN DE UN WEB SERVICE UTILIZANDO PLATAFORMA J2EE

El sistema implementado en el capitulo anterior se basa en plataforma J2EE lo cual brinda una serie de posibilidades tecnológicas donde visto en capitulo anteriores se presenta en

1. Tecnología Web que permite definir una interfaz gráfica: Servlets, JSP y JSTL.
2. Componentes EJB (Enterprise java Beans). Entity Beans y Session Beans.

Por lo que se puede establecer que la aplicación trabaja en un modelo en tres capas, es decir tenemos una serie de programas clientes, tanto clientes Web, clientes java, y el contenedor o servidor esta constituido de aplicaciones Web que viene ser la interfaz grafica de la aplicación en particular y un servidor EJB orientado a la comunicación con el servidor de base de datos o conocido también como (back end en inglés) donde cabe la lógica del negocio.

Arquitectura de una aplicación Web con un contenedor Web y un contenedor de EJB (quizás de distinto fabricante). Si los dos contenedores corren en la misma máquina física, se puede considerar como una arquitectura en tres capas.

Esquema Web Service

Con esta descripción vemos que las empresas no requieren cambiar varios aspectos de sus aplicaciones ya que el desarrollo de ciertos módulos de la aplicación en Web Services viene a conformar una opción que no implica realizar cambios que interfieran en el funcionamiento o crecimiento de la aplicación, también no requiere de cambiar ni es un esquema que pueda estar separado de la arquitectura cliente servidor o bien que en este caso es en tres capas basado en la tecnología J2EE.

Simplemente la idea de utilizar en la aplicación es un soporte del API de SOAP esto no involucra un desarrollo o pensar en un esquema diferente, ya que el propósito es incluir que ciertos métodos trabajen de forma independiente para que posteriormente interactúen con otro sistemas, manejando un estándar : comúnmente también sobre la capa HTTP como puede observar en la figura siguiente.

4.5.2. OPCIONES A J2EE DONDE UNA APLICACIÓN PUEDE SOPORTAR LA IMPLEMENTACIÓN WEB SERVICE

.NET

- Define un Common Language Runtime (CLR) y un IL (Intermediate Language) al que todos los lenguajes conformes a .NET compilan
- Idea similar a la máquina virtual de Java y a los bytecodes generados por el compilador de Java, respectivamente
- Lenguajes Visual Basic .NET, Visual C++ .NET, Visual C# .NET, Visual J# NET, etc.
- Tecnologías ADO.NET, ASP.NET, COM+: similares en concepto a JDBC, JSP y EJB, respectivamente
- Son una mejora de sus versiones anteriores (ADO, ASP, COM, etc.)
- APIs para XML
- Implementaciones Principalmente la de Microsoft
- Mono (Open Source, Linux y otros sistemas operativos):

LAMP

- Uso de de Linux , Apache , MySQL , Perl/PHP/Python/Perl/PHP/Python
- Lenguajes tipo Script
- Acceso a base de datos
- Tecnologías Web
- Soporte para XML
- Requiere menos conocimientos técnicos que J2EE o .NET
- Cada lenguaje de programación maneja módulos que contienen y soportan SOAP para creación y despliegue de Web Service.: NuSOAP para PHP, SOAP: Lite para Perl, etc.

LOS ASPECTOS BÁSICOS PARA EL DESARROLLO VIABLE DE UN WEB SERVICES A TRAVÉS DE LAS TECNOLOGÍAS. J2EE FRENTE A .NET SON:

Los Web Services conciernen con cuatro retos básicos

- Descripción del servicio(Service Description)
- Implementación del Servicio (Service Implementation)
- Servicio de Publicación , Descubrimiento y Unión(Publishing,Discovery and Binding)
- Servicio de Invocación y de Ejecución(Service Invocation and Execution)

Descripción del Servicio

Para la descripción del servicio hace uso hace uso del lenguaje Web Service Description Language(WSDL) el punto básico para que un Web Service se describa en un modo que necesite definir una gramática XML para descubrir los Web Services como una colección de mensajes o endpoints.

J2EE

Los Web Services se interconectan usando WSDL para convenir en un acuerdo en el propio formato para cada transferencia de documentos XML. Una tercera parte del Web Service quien quiere realizar transacciones de negocios con una compañía que manipula Web Services con J2EE tienen que buscar información del Web Services de la compañía a través de un registro

.Net

Como J2EE Web Services un Web Services .Net soporta la especificación WSDL 1 y utiliza un documento WSDL para describirse así mismo.

.Net provee componentes del lado del cliente que permiten que una aplicación invoque operaciones en Web Services descritas por un documento WSDL y un componente del lado del servidor que mapea las operaciones de los Web Services con métodos de llamada COM-Objects como es descrito un WSDL y un archivo Web Service Meta Language(WSML). Este archivo es indispensable para la implementación de SOAP en Microsoft.

Implementación del Servicio

La implementación significa estructurar datos y operaciones dentro de un documento XML que compila con la especificación SOAP. Una vez que un componente de un Web Service es implementado, un cliente envía un mensaje SOAP como un documento XML y el componente regresa el documento XML cliente como respuesta.

J2EE

Existen clases Java y aplicaciones que pueden ser cubiertas utilizando una API Java para XML basada en un JAX-RPC(JAX-RPC) y es expuesta como Web Services.

JAX-RPC usa XML para hacer llamadas a procedimientos remotos que exponen un API por ordenamiento (empaqueta parámetros y regresa valores para ser distribuidos) y descarta argumentos para transmitir y recibir procedimientos de petición remotos.

Con J2EE los servicios de negocio como Enterprise Java Beans son cubiertos y expuestos como Web Services . El resultado de este empaquetamiento es un Web Services a través de SOAP que conforma una interfase WSDL con base en métodos EJB. La Arquitectura J2EE Web Services es un (conjunto de XML) panorama basado en XML, y provee infraestructura que permite a las compañías integrar una lógica de negocios.

Punto NET
Las aplicaciones .Net no se ejecutan en un código nativo de máquina. Todos los programas son interpretados en un código binario llamado Microsoft Intermediate Language (MSIL). Este código portable, binario es compilado después a un código nativo usando un compilador Just in Time (JIT) en tiempo de ejecución y corre en una máquina virtual llamada Common Language Runtime (CLR). Este modo es similar al que trabaja Java, excepto que .Net abarca varios lenguajes. Con la plataforma .Net Microsoft provee varios lenguajes basados en Common Language Infraestructure (CLI), tal es como C++, VB.Net C# etc.

Publicación, Descubrimiento y Enlace

Una vez que los servicios están implementados deben publicarse y descubrirse para ellos se utiliza un registro como medio primario. Los Web Services contienen la infraestructura de datos y taxonomías utilizadas para descubrir un Web Services y los proveedores de Web Services. Un registro pude ser instalado por organizaciones privadas o por una tercera parte neutral.

J2EE

Sun Microsystems se posiciona con su API XML Registers (JAXR) para que interopere con varios tipos de registros.
JAXR permite que sus clientes acceder a un proveedor de Web Services.

Hay tres tipos de proveedores JAXR:

- EL JAXR Pluggable Provider(es independiente del tipo de registro)
- El Register specific JAXR Provider
- El JAXR Bridge Provider, el cual no es especifico para un registro en particular. Sirve como puente para una clase de registros tal es como ebXML o UDDI.

Punto Net

Utiliza para el descubrimiento de un Web Service un archivo de descubrimiento llamado (DISCO) .Un archivo de publicación DISCO es un documento XML que contiene enlaces a otras fuentes que describen el Web Service. A pesar de la adaptación que se tiene con UDDI, Microsoft lo ha soportado con la finalidad de maximizar la interoperabilidad.

Servicio de Invocación y Ejecución

El Protocolo Simple Access Protocolo (SOAP) define un cuadro o esquema de mensajes para intercambiar datos estructurados e información tipificada a través de la Web

J2EE

Utiliza una llamada a procedimientos remoto JAC (JAX-RPC) para enviar métodos SOAP a diferentes partes remotas y recibir los resultados JAX-RPC habilita a los desarrolladores Java construir Web Services incorporando XML basado en RPC de acuerdo a la especificación SOAP 1.1

Una vez que un servicio JAX-RPC ha sido definido e implementado, el servicio se despliega en un servidor JAX-RPC de un sistema en ejecución. El paso para desplegar depende del tipo de componente que se haya utilizado para implementar el servicio. Por ejemplo un servicio basado en un Enterprise Java Bean (EJB) se despliega en un contenedor EJB.

Durante el despliegue de un servicio JAX-RPC la herramienta de despliegue configura uno o mas protocolos conectados a este servicio JAX-RPC. Una conexión une la definición de servicio abstracta a un protocolo específico basado en XML.

Un cliente Web Services usa JAX-RPC para invocar métodos remotos en un puerto de servicio descrito por un documento WSDL.

Punto NET

En el esquema de Microsoft Punto Net, las partes interesadas pueden acceder a un Web Services a través de la implementación de un receptor Web Service. Para esto un sistema necesita entender los mensajes SOAP, generar respuestas SOAP, proveer un contacto WSDL para el Web Service, y anunciar el servicio vía UDDI
Los desarrolladores de Punto NET receptores y clientes Web Services, basados en SOAP presentan tres opciones:

1. Utilizar clases de mensajes Punto NET SOAP.

2.- Construir un receptor Web Service manualmente, usando MSXML. ASP, o ISAPI.

3.- Utilizar el Microsoft Soap Toolkit versión 2 para construir receptores Web Service que conecta a un negocio.

Características de las posibles arquitecturas a utilizar:

Teniendo un panorama de cómo se maneja un Web Services a través de J2EE y Punto NET se tiene ventajas y desventajas.

Una de las ventajas de usar Punto NET para Web Services, es que la arquitectura ha sido diseñada para ese propósito, mientras que J2EE se ha retroalimentado con la conjunción de nuevas API's.

Una ventaja de usar J2EE es que se tienen diversas opciones de versiones para software comerciales y proyectos de licencia libre los cuales utilizan esta arquitectura, específicamente servidores de aplicación. Es importante mencionar que los servidores de aplicación J2EE con licencia libre se apegan a los estándares que provienen de SUN.

Otra diferencia que se encuentra es que Punto Net esta ligado solo a un sistema operativo en este caso Windows, así que no se necesita instalar por separado software de servidores de aplicación. Con Java se tiene la opción de que es multiplataforma, a través de varios distribuidores de servidor de aplicación J2EE que han agregado extensiones propias que los distingue de los competidores.

Punto Net ofrece soluciones rápidas, en tanto la especificación EJB tiene una larga escala de proyectos de desarrollo que refuerzan mucho mejor las prácticas de código y reglas de diseño.

Así también Punto Net ofrece un desarrollo y despliegue muy fácil, sin embargo esto requiere de una cotización mas alta si se requiere construir un entorno con una misión critica en que la confiabilidad, la integridad de transacción y la administración de mensajes son esenciales. Por el contraste los proyectos J2EE requieren de una cotización menos alta para comenzar con ello.

Hay que mencionar que Microsoft tiene un gran soporte en lo que respecta a Web Services, J2EE requiere de actualizar las versiones de las API.

Para concluir la selección de que software utilizar, depende de estos aspectos, si se cuenta con un conocimiento de programación en Java y se desea presentar la aplicación en muchos dispositivos, J2EE es lo que se necesita. Por el contrario si los conocimientos de programación o código existente esta basado en una variedad de mensajes y si deseas desplegar tu aplicación en un entorno Windows, Punto Net es la opción.

Servidores de Aplicación en el mercado basados en J2EE y Punto Net
La mejor interfase, excelente clustering y alta disponibilidad de características, líder en desarrollo de herramientas para Web Services desde los Enterprise java Beans (EJB), lo presenta Web Logic Server 8.1. Web Logic considerado un producto con un alto nivel de configuración, despliegue, escalabilidad disponibilidad, administración, interoperabilidad y documentación

Cabe mencionar la gran mención que se realiza para IBM WebSphere Application Server 5, el cual encaja con Web Logic en la administración de interfases y clustering. Se reconoce el esfuerzo de IBM por retirar de su configuración el esquema que usa para su base de datos DB2 a favor de la configuración estándar XML.

4.6 IMPLEMENTACIONES CON SOFTWARE DE LICENCIA LIBRE (OPENSOURCE)

El proyecto ha partido de un enfoque para analizar, desarrollar e implementar una aplicación basada en J2EE un WebServices, el proyecto XML Apache se encuentra en un momento donde puede competir en funcionalidad a otros de tipo comercial a su vez es el proyecto apache un recurso para componentes de organizaciones líderes tal es el caso de IBM, y Sun Microsystems que han realizado a su vez cubiertas aportaciones a proyecto de software libre y su serie de herramientas que maneja la fundación Apache .

Así también es importante señalar hoy en día el auge que muestra el uso de XML , por ello, el proyecto XML la fundación de Software Apache, consiste en un conjunto de subproyectos enfocados a distintos aspectos XML.

Xerces: Segmentador de componentes de texto(parser) en Java y C++(más uniones de Perl y COM)

Xalan: Procesador de hojas de estilo XSLT, en Java y C++

Cocoon:Publicación Web basada en XML en Java

AxKit: Publicación web basada en XML en modo perl.

FOP: Objetos de formato XSL en Java

Xang: Desarrollo rápido de páginas dinámicas de servidor en JavaScript

SOAP: Protocolo de Acceso a Objetos Simple

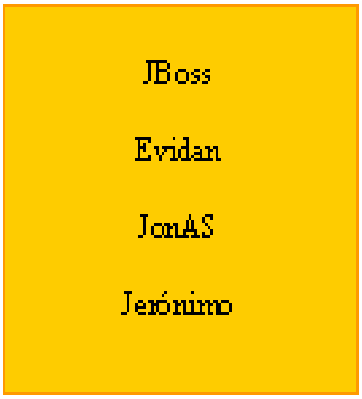
Crimson: Un segmentador de componentes de texto basado en XML

Herramientas para el desarrollo Webservices

1. **-Axis (Apache eXtensible Interaction System)** o bien SOAP 3.0 pero Axis contiene la implementación SOAP y presenta mas opciones que complementan las características para una implementación del WebServices.
2. -**Servidor de Web Java Jakarta Tomcat de Apache** , Tomcat es un recipiente de servlets utilizando el desarrollo de tecnologías Java Servlet 2.3 y Java ServerPages 1.2 para ello se ha encontrado a Jboss que emplea en sus versiones el uso de Tomcat y que este mismo es específicamente un contenedor EJB aunque hace uso de Jetty como una opción de servidor web .
3. **-Segmentador de componentes de texto XML Xerces de Apache lo que Axis** ya contiene este segmentador .

El proyecto en software libre considera requisitos como son:

- Establecer un hardware adecuado que soporte los requerimientos mínimos de memoria y procesador.
- Documentación de las características del Servidor, configuración que soportan el servicio de Web Service.
- Independiente de plataforma.
- Implementación en J2EE y LAMP(Linux, servidor Apache, lenguajes perl, PHP, Python)
- Herramientas del grupo de licencia libre más importante como Apache.
- Integración con entornos de desarrollo o bien herramientas IDE que facilitan también el proceso de desarrollo de la aplicación como puede ser Eclipse
- Uso de la herramienta **ANT** que también facilita el proceso de desarrollo, implementación, compilación y despliegue de la aplicación entre otras funciones que pueden incluir con la finalidad de reducir el tiempo de procesos que interviene en la puesta en marcha del proyecto.



Servidores de aplicación con licencia libre que soportan WebSerrvices basados en J2EE

Si una aplicación sólo usa las APIs estándares es posible instalarla sobre cualquier servidor de aplicaciones conforme a J2EE no dependen de un fabricante.

Otras herramientas adicionales que pueden facilitar el procesos de desarrollo son conocidas como herramientas IDE estas brindan intefases gráficas de las carpetas del proyecto, provee opciones para despliegue, compilación y a su vez incluir otras opciones orientas a disminuir y facilitar ciertas actividades que intervienen en el desarrollo de aplicación y por ende de un WebServices,

Mientras un JDK SDK ofrece las herramientas para compilar y ejecutar programas en Java éste no ofrece un ambiente de trabajo para *proyectos complejos*, esto es, si se compila una o dos clases quizás el comando javac ofrecido en los JDK es suficiente, pero si un proyecto estará compuesto por 100 o 200 clases javac sería muy deficiente. Los IDE's ("Integrated Development Environment") ofrecen un ambiente gráfico en los que se tiene acceso a mayor número de herramientas no ofrecidas en los JDK's: Debuggers más elaborados, check-points dentro de la compilación, creación de WARS(Web-Archives), "Wizards" , entre otras cosas. Algunos IDE's son:



Herramientas IDE

Se enumeran servidores de arquitectura J2EE ya que existen diversos fabricantes los cuales han diseñado sus servidores de aplicación o contenedores en componentes de esta tecnología.

Actualmente trabajar con JBoss parece ser un servidor de software libre con un grupo de desarrollo diverso cuyos antecedentes parecen tener características semejantes a J2EE, cuenta con servidor Web Tomcat o Jetty y un servidor EJB. Así también se aprecia que la mayoría maneja un solo lenguaje como es java, este no considera ningún problema ya que es un lenguaje multiplataforma.

JBoss progresa en sus versiones, y a su vez contiene configuraciones diferentes que permiten trabajar módulos de forma independiente.

Es un servidor que ha mostrado un rendimiento aceptable, sencillo de instalar y configurar. Así también permite la integración de otras bases de datos y trabajar Web Services ya que contiene a JBoss-net o JBossWS, o bien, se tiene la opción de trabajar Axis de forma separada

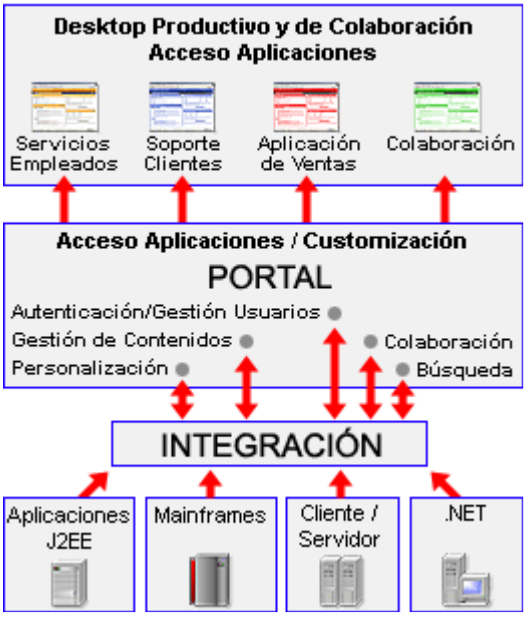
4.7 ESQUEMA WEB SERVICE COMO UNA APLICACIÓN DE NEGOCIO

En la figura anterior plantea la aplicación real y el objetivo tecnológico como de negocio de tener una serie de Web Services de varios sistemas trabajando como módulos independientes pero siendo parte integral de una aplicación en particular .Aquí se puede ver que tanto un cliente Web Service solo requiere manejar XML, SOAP sobre HTTP.

Con lo anterior se puede pensar que una aplicación que trabaja en una plataforma o esquema J2EE presenta una mejor gama de posibilidades ya que se pueden utilizar diferentes servidores de aplicación de diversos fabricantes y de licencia libre.

El Web Service se ha planteado como un tecnología de integración de aplicación basando en un mejor éxito y aceptación superando a un antecesor como es CORBA, el cual se ha estancado, y a superado esta idea principalmente por que Web Service maneja un estándar basado en XML, que es lenguaje bien soportado por varios fabricantes, inclusive de grupo de desarrollo de software libre.

Así mismo en esta tabla es importante apreciar algunas características para conocer ciertas diferencias y ver cuales de esas hacen posible que Web Service tiende a ser una tecnología en proceso de desarrollo para mejorar la integración de las aplicaciones.



Integración posible a través de un Web Service

El rápido desarrollo de los estándares (XML, ebXML, SOAP, WSDL y UDDI) está consolidando la plataforma de Web Services como la definitiva a la hora de integrar aplicaciones empresariales.

Siendo Web Service una opción que puede pensarse en diferentes escenarios y diferentes plataformas debido al aporte de fabricantes se puede hacer presentes su desarrollo a parte de J2EE en tecnología .NET o en LAMP (Linux, servidor Apache, Perl, Mysql). Cabe señalar que no es importante conocer simplemente características y saber la gama de posibilidades en las cuales puede un sistema de cierta empresa hacer uso y obtener un potencial, también es considerar sus necesidades y mejoras en sus procesos.

Los puntos a considerar del proyecto de implementación de un Web Service con software libre, es primero descubrir los alcances de su aplicación.

Considerar el software que provee las compañías comerciales y comparar con base a la implementación de un Web Service visto en el capitulo anterior como un parámetro donde se analice sobre el comportamiento del mismo, analizar en que forma se produce su instalación , prueba en marcha, documentación disponible, escalabilidad algunas opciones alternativas de software etc.

Los Web Services crean **compatibilidades e interoperabilidad** entre diversas aplicaciones Web empaquetadas y personalizadas, de una manera estandarizada y neutral a nivel de proveedores de sistemas de información. Los Web Services tienden a ser una alternativa más económica y benéfica cuando se usan dentro de un escenario de negocios de nivel empresarial que combina las funcionalidades de múltiples aplicaciones con servicios empresariales de fácil uso. Dicho escenario de negocios demanda un enfoque arquitectónico orientado hacia los Web Services.

Los Web Services desempeñan un papel preponderante dentro del concepto de Arquitectura de Servicios Empresariales**.** Los servicios empresariales exponen las funcionalidades y los datos de las aplicaciones para que éstos puedan ser accedidos por cualquier usuario de servicio. Así como los Web Services eliminan la complejidad de la conectividad de las plataformas, así mismo los servicios empresariales eliminan la complejidad de la integración de las aplicaciones.
Los Web Services tienden a ser un innovación frente a los sistemas abiertos, comunicación de aplicaciones, pero lo importante es establecer que negocio o empresa debe considerar su situación , plantear sus necesidades, si dentro de sus necesidades encuentra el mejorar procesos , el hacer peticiones y respuestas de clientes de forma directa con proveedores ya que resulta una alternativa atractiva. Pero la cuestión es si el implementar esta opción a la empresa realmente bajará mi costo de desarrollo puesto que no se tienen que invertir en programación elaborada simplemente el sistema implementara ciertos módulos capaces de ser descritos en la pagina WSDL, para ser publicados bien en un repositorio UDDI.

Transformar la infraestructura de tecnología de Información, orientada hacia los servicios, a través de una serie de pasos manejables y eficientes.

Una de las aplicaciones donde empresas hoy en día están encontrando soluciones a procesos internos, por ejemplo, un portal empresarial puede usar Web Service para ofrecer interfases basadas en roles que permite a los empleados trabajar mancomunadamente. La plataforma de aplicaciones ofrece funcionalidades de aplicaciones basadas en java, en calidad de Web Service. Estos servicios que describen mediante el empleo del Lenguaje de Descripción de Servicios Web (Web Services Description Languae-WSDL)

Los Web Services han sido idealizados para ofrecer a través de una estructura usada por el intermediario de integración para describir las interfaces de WSDL en un repositorio UDDI (siglas en inglés para Descripción, Descubrimiento e Integración Universales). La administración de procesos de negocios coordina las actividades de los WebServices prestados por los socios de negocios para la administración de los procesos a través de las aplicaciones.

Existe otras tecnología que las empresas han hecho uso para el desarrollo de sus negocios electrónicos , muchas empresas de software utilizan su propias plataformas , es decir , contamos Visual Studio .Net que solo puede ser utilizado en una plataforma que maneje un sistema operativo Windows y un servidor de aplicación exclusivo de Microsoft como puede ser Windows 2000 server , así como una base de estos mismo,

por otro lado, otra empresa decide de acuerdo a sus necesidades y alcances particulares el desarrollo de sus aplicaciones en un sistema operativo Unix cuyo servidores de aplicaciones puede ser WebSphere de IBM , WebLogic de BEA, Sun application Server de Sun Microsystems entre varios basados en tecnología J2EE.

Lo más importante que una empresa debe considerar con esta alternativa tecnológica es proveer una innovación de la misma, obtener las aplicaciones y ventajas que le va dejar el uso de un Web Service.

Como primer punto la empresa debe considerar:

La productividad, la seguridad, la economía, la innovación, rentabilidad.

La productividad que se traduce en los objetivos inmediatos logrados con eficiencia y eficacia, las empresa necesitan crecer y parte de sus sistemas es ofrecer un funcionamiento adecuado que satisfaga las necesidades particulares: por ejemplo, si la empresa quiere a dar a conocer un producto o servicio la empresa puede utilizar la alternativa de Web Service para publicar estos mismos en un registro UDDI que es similar al funcionamiento de paginas amarillas donde las empresas buscan localizar e invocar que empresas han publicado módulos de sus sistemas y de esta forma se puede apreciar un escenario en el cual los sistemas de las empresas logren hacer alguna petición de productos y proveer de los mismos independiente de la plataforma en la cual se encuentren, Esto confirma la diferencia de un manejo en tres capas o cliente servidor donde la comunicación se basa vía Internet al protocolo TCP/IP , el Web Service tiende a satisfacer una necesidad en la cual permite la comunicación vía el protocolo SOAP y la facilidad de que los sistemas se intercomuniquen. Lo importante de una empresa es que si este escenario realmente es productivo, mas eficiente, que utilizar otro.

En cuanto al software libre existe una serie de proyectos muy bien soportados, con diversa documentación. Hablando precisamente del grupo de herramientas Apache Axis que viene a ser el motor SOAP con todas las especificaciones de este protocolo.

La arquitectura brinda componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

JBoss tiene la ventaja de contener un modulo orientado a Web Service, llamado JBoss-Net y JBossWS para la versión mas reciente. Esta se basa en Axis. Dicho software tienen un código accesible con documentación sumamente completa, tanto instalación, implementación de ciertas clases y protocolos específicos para la realización del Web Service, lo importante es que JBoss mantiene este modulo y ya no requiere de su instalación.

4.8 WEB SERVICES SEGUROS

Discusión sobre la seguridad de los servicios Web XML. En principio no son seguros, pero se pueden hacer cosas para evitar accesos no permitidos. Teniendo en cuenta todos los aspectos de seguridad, autenticación y autorización, confidencialidad e integridad de datos, etc., y el hecho de que la especificación SOAP ni siquiera menciona la seguridad, es fácil llegar a la conclusión de que la respuesta es negativa. Pero no hay que subestimar los servicios.

Hoy en día, se pueden tomar muchas medidas para crear servicios Web XML seguros. A la hora de tratar la seguridad en servicios Web XML, es necesario considerar los siguientes puntos:

Limitar el acceso a servicios Web XML a usuarios autorizados, evitar que los mensajes puedan ser visualizados por personas no autorizadas, etc.

Atacar el objetivo mediante Red, capa de transporte, sistema operativo, servicio o aplicación.

Definir el nivel de interoperabilidad deseado y que necesitamos en nuestra solución: Local o global.

Establecer como se puede aumentar la seguridad de los Web Services actuales.

Por ejemplo, un servidor de seguridad puede funcionar con servicios Web Services para limitar el acceso a ciertas funcionalidades (métodos) basándose en la identidad del cliente y las directivas definidas para cada cliente.

Creación de una infraestructura segura una infraestructura segura es la clave para contar con Web Services

Crear una estrategia de implementación planeada con detenimiento para integrar las medidas de seguridad en el conjunto de una red empresarial, basándose en los dos puntos anteriores.

Creación de conexiones seguras Uno de los métodos más sencillos para aumentar la seguridad de los Web Services consiste en asegurarse de que la conexión entre el cliente Web Service XML y el servidor es segura. Esto se puede llevar a cabo mediante varias técnicas, dependiendo de la extensión de la red y el perfil de actividades de las interacciones. Tres de las técnicas más comunes y accesibles son: reglas basadas en un servidor de seguridad, SSL (Secure Sockets Layer) y redes privadas virtuales (VPN. Si sabe con exactitud qué equipos van a tener acceso a los Web Services, puede utilizar reglas de servidor de seguridad para restringir el acceso a equipos con direcciones IP conocidas. Esta técnica resulta particularmente útil si desea restringir el acceso a equipos en una red privada virtual (por ejemplo, una red LAN/WAN corporativa) y no desea mantener el contenido de los mensajes en secreto (cifrados. Los servidores de seguridad,como **JBoss** contiene módulos de seguridad que pueden proteger a la aplicación: **JAAS**(Java Authentication and Authorizarion Service), pueden ofrecer reglas avanzadas basadas en directivas que proporcionen distintas restricciones para clientes diferentes según su origen o identidad. Esto resulta de gran utilidad cuando se espera que determinados clientes tengan acceso a funcionalidades (métodos) diferentes de los mismos Web Services. Se puede utilizar **SSL** para establecer conexiones seguras en redes que no son de confianza (como Internet. SSL cifra y descifra mensajes enviados entre el cliente y el servidor. Al cifrar los datos, se evita que los mensajes sean leídos por terceros durante la transmisión. SSL cifra el mensaje del cliente y, a continuación, lo envía al servidor. Una vez que el servidor lo recibe, SSL lo descifra y comprueba que procede del remitente correcto (proceso que se denomina autenticación. El servidor, o el servidor y el cliente, pueden tener certificados que se utilizan como parte del proceso de autenticación y que proporcionan capacidades de autenticación además del cifrado de la conexión.

Actualmente, los WebServices están siendo ampliamente aceptados por las empresas para el desarrollo de software de uso interno. De este modo, los servicios pueden implementar toda su funcionalidad y permanecer seguros tras el Cortafuegos de la compañía(firewalls). Los desarrollos actuales no ayudan a la cooperación entre las empresas ya que no hay ningún estándar establecido sobre las técnicas de seguridad.

Debido a la tecnología que es usada por los WebServices, y en concreto al uso de SOAP, las técnicas de seguridad convencionales que se han venido usando en Internet, ya no son suficientes. Con SOAP, cada mensaje simple que se intercambia realiza múltiples saltos y es rutado a través de numerosos puntos antes de que alcance su destino final. Es por ello que los WebServices necesitan tecnologías que protejan los mensajes desde el principio hasta el final.

Existen otro conjunto de técnicas adicionales que se pueden usar para garantizar la seguridad a nivel de mensaje. Estas son:

- **Encriptación XML:** Evita que los datos se vean expuestos a lo largo de su recorrido.
- **Firma Digital XML:** Asocia los datos del mensaje al usuario que emite la firma, de modo que este usuario es el único que puede modificar dichos datos.
- **XKMS y los Certificados: XKMS (XML Key Management Specification**) define web services que se pueden usar para chequear la confianza de un certificado de usuario.
- **SAML y la Autorización: SAML (Security Assertion Mark-up Language)** hace posible que los web services intercambien información de autentificación y autorización entre ellos, de modo que un web service confíe en un usuario autentificado por otro web service.
- **Validación de datos:** Permite que los web services reciban datos dentro de los rangos esperados.

Además, también hay técnicas que permiten mantener la seguridad a otros niveles. La seguridad en UDDI permite autentificar todas las entidades que toman parte en la publicación de un WebService: proveedor, agente y consumidor del servicio. De este modo, nadie podrá registrar servicios en el papel de un proveedor o hacer uso de ellos sin contar con los permisos adecuados.

Algunas de las empresas más importantes en el desarrollo de Negocio Electrónico como IBM, Intel, Microsoft u Oracle, han creado el WS-I: organización para la Interoperabilidad de los Web Services. El objetivo de dicha organización es la promoción de la estandarización de los web services de modo que se fomente la cooperación e interoperabilidad entre las compañías y mercados.

Algunos ejemplos

Las principales compañías del mundo han empezado a desarrollar soluciones mediante la tecnología de los web services. Algunos ejemplos son:

- **Microsoft:** Recientemente ha anunciado la disponibilidad de su primer WebService, llamado MapPoint .Net. Mediante este servicio, el usuario podrá conocer su localización exacta y otros datos adicionales relacionados con su posición actual, como información de tráfico, rutas posibles o puntos comerciales cercanos.

- **IBM:** Ha implementado una solución basada en los web services llamada e-Business sobre demanda. Esta solución permite la construcción de Extranets que ayuden a las empresas a ver los catálogos de productos, realizar y localizar pedidos o chequear el estado del inventario en tiempo real.

- **Líneas Aéreas Escandinavas:** Estas líneas aéreas han desarrollado un servicio web que permite a los usuarios comprar billetes y chequear el estado de los vuelos, mediante el uso del teléfono móvil.

Aunque la idea de la programación modular no es nueva, el éxito de esta tecnología reside en que se basa en estándares conocidos en los que ya se tiene una gran confianza, como el XML. Además, el uso de los web services aporta ventajas significativas a las empresas. El principal objetivo que se logra, es la interoperabilidad y la integración. Mediante los WebServices, las empresas pueden compartir servicios software con sus clientes y sus socios de negocio. Esto ayudará a las compañías a escalar sus negocios, reduciendo el coste en desarrollo y mantenimiento de software, y sacando los productos al mercado con mayor rapidez. La integración de aplicaciones hará posible obtener la información demandada en tiempo real, acelerando el proceso de toma de decisiones. La evolución de Internet hacia los WebServices, mejorará los resultados globales de las empresas, reduciendo sus gastos y guiándolas hacia una mejora progresiva de la calidad. La adopción de la tecnología de servicios web por la industria es el primer paso hacia una economía global.

4.9 RESUMEN DE VENTAJAS Y DESVENTAJAS DE LOS WEB SERVICES UTILIZANDO SOFWARE LIBRE

Ventajas del uso de los WebServices

Por otro lado las empresas buscan un mecanismo de incrementar su negocio a través de un medio electrónico, masivo y poderoso como lo es Internet. Hoy en día se busca evolucionar en los sistemas, esto se encuentran obsoletos, poco flexible, es decir, se pretende encontrar la interoperabilidad de los sistemas, la localización de los negocios electrónicos.

Se tiene que dar soluciones a los negocios sobre el adecuado uso de la tecnología de los Web Services como parte esencial en el crecimiento del negocio, la versatilidad de sus sistemas, el rendimiento, mantenimiento, desarrollo e investigación por lo cual tienen que innovar sus sistemas.

Un benéfico que se ve en esta tecnología es el uso de un estándar como lo es el lenguaje de marcación XML el cual es la base donde se constituye el desarrollo de protocolos y archivos que permiten unificar el concepto de Web Services.

Este estudio esta enfocado a conocer si esta tecnología puede ser adecuada y propicia utilizando un desarrollo con licencia libre, como consecuencia determinar si es viable la implementación o el uso de aplicaciones a través de un Web Service utilizando software libre relacionándolo al negocio electrónico donde las empresas buscan encontrar en ello una opción que les brinde la rentabilidad de sus negocios, ya que la forma mas adecuada es la captación de clientes y por supuesto la venta y promoción de sus productos o servicios, el servidor de JBOSS ofrece confiabilidad, seguridad , versatilidad, robustez en el desarrollo de software libre, como puede encontrarse en un software de tipo comercial.

Beneficios de negocio:

- Disminuir el tiempo de realización de transacciones (time to market) para desplegar tecnologías. WebServices significa integrar o tener antes disponible las soluciones de negocio. El uso de esquemas y patrones replicables, libera a los desarrolladores de tareas repetitivas y les permite focalizarse en la lógica de negocio.
- Disminuir el coste de propiedad, utilizando una infraestructura integrada, segura, escalable y disponible.
- Proveer la flexibilidad para integrarse con productos de otros fabricantes que estén basados en estándares abiertos.

Las ventajas pueden basarse de acuerdo al análisis de un escenario donde pueda ser aplicado un Web Service, Si la finalidad es que una empresa promueva de cierta forma sus productos y servicios logrando con Web Service una mejor presencia para el giro de su empresa. Pero antes que nada debe definirse el propósito de la empresa, si realmente su esquema requiere de estar en contacto inmediato y efectivo con clientes y proveedores que facilite el proceso de ventas o de algún servicio en particular. También es importante considerar la inversión que involucra una implementación del sistema asi como que utilidades que produce esta tecnología y saber si se requiere innovar o hacer modificaciones que no resulten demasiado costosas.

- Exponer método de una aplicación a Web Service disponibles a otros sistemas de otras plataformas, por lo cual la intención de este esquema es que diversas empresas no requieren adquirir diferente software, desarrollar complejas aplicaciones, o bien migrar sistemas a otras plataformas para obtener un resultado como éste**.**

Lo mas importante es que se emplea un estándar de comunicación lo que permite la interoperabilidad de las aplicaciones lo cual cumple con la idea de computadora distribuida.

- El uso de software libre permite estudiar un camino o una opción de solución que puede una empresa adquirir para **minimizar costos** ya que invertir en cambiar un sistema a otra plataforma requiere de tiempo de desarrollo, adquisición de software, programación, si el único objetivo es exponer ciertos métodos que puedan ser compartidos y tener acceso a protocolos.
- Una alternativa de desarrollo para la **empresa de proponer un mecanismo adicional de promoción de productos y servicios.** La finalidad es exponer los sistemas de las empresas e interactuar con los módulos publicados. La información es procesada por mensajes SOAP donde ya existe un estándar que facilita la obtención de un sistema distribuido y abierto.
- Innovación tecnológica capaz de distribuir los sistemas
- Las empresas **no tiene que emigrar sus sistemas a otras plataformas** para lograr un objetivo de comunicación directa con otra aplicación implantada en otra plataforma

- Existen alternativas hoy en día de lograr el empleo de software libre y comercial que pretende decrementar el costo
- Con software libre se tiene la capacidad de establecer condiciones adecuadas puesto que se tiene acceso al código lo cual permite una implementación mas flexible.
- Competencia en cuanto a la funcionalidad de un WebServices a través de Apache Axis y en conjunto con JBoss se pueden realizar cosas importantes y adaptarlas a un sistema en particular.Como se ha visto en el capítulo anterior éste servidor tiene la capacidad de implementar el WebService, porque esta basado en J2EE lo cual hace posible tener una aplicación con dicha arquitectura y que hoy en día muchas empresas manejan este modelo. Esto ha permitido interactuar con la plataforma totalmente distinta como es Microsoft Visual Studio :Net., es un ejemplo muy claro que el protocolo SOAP permite la comunicación, pero es necesario señalar que la infraestructura que existe detrás tiene un objetivo el cual es obtener los mejores rendimientos en los procesos y actividades involucrados en los sistemas de las empresas y sistemas de información.

SOAP es un protocolo elaborado para facilitar la llamada remota de funciones a través de Internet, permitiendo que dos programas se comuniquen de una manera muy similar técnicamente a la invocación de páginas Web.

El protocolo SOAP tiene diversas ventajas sobre otras maneras de llamar funciones de manera remota como DCOM, CORBA o directamente en TCP/IP:

- Es sencillo de implementar, probar y usar.
- Es un estándar de la industria, creado por un consorcio del cual Microsoft, IBM, SunMicrosystems. HP, etc, forma parte, adoptado por W3C (http://www.w3.org/TR/SOAP/) y por varias otras empresas.
- Utiliza los mismos estándares de la Web para casi todo: la comunicación se hace mediante HTTP con paquetes virtualmente idénticos; los protocolos de autenticación y encriptación son los mismos; el mantenimiento de estado se hace de la misma forma; se implementa normalmente por el propio servidor Web.
- Atraviesa "firewalls" y routers, que "piensan" que es una comunicación HTTP.
- Tanto los datos como las funciones se describen en XML, lo que permite que el protocolo no sólo sea más fácil de utilizar sino que también sea muy sólido.
- Es independiente del sistema operativo y procesador.
- Se puede utilizar tanto de forma anónima como con autenticación (nombre/clave).

Las solicitudes SOAP se pueden hacer en tres estándares: GET, POST y SOAP. Los estándares GET y POST son idénticos a las solicitudes hechas por navegadores de Internet. SOAP es un estándar similar a POST, pero las solicitudes se hacen en XML y permiten recursos más sofisticados, como pasar estructuras y arreglos ("arrays").

Independientemente de cómo se haga la solicitud, las respuestas siempre son en XML. XML describe perfectamente los datos en tiempo de ejecución y evita los problemas ocasionados por cambios inadvertidos en las funciones, ya que los objetos llamados tienen la posibilidad de validar siempre los argumentos de las funciones, haciendo que el protocolo sea muy sólido.

Así mismo, SOAP define un estándar llamado WSDL, que describe perfectamente los objetos y métodos disponibles a través de páginas XML accesibles por la Web. La idea es la siguiente: quien publica un servicio, crea también estas páginas. Quien quiera llamar el servicio, puede utilizar estas páginas como

"documentación" de la llamada y también utilizarlas antes de llamar las funciones para verificar si cambió algo.

SOAP se puede implementar fácilmente en casi cualquier ambiente de programación. Actualmente, existen diversos paquetes de desarrollo SOAP para diversos sistemas operativos y lenguajes de alto nivel.

Desventajas del uso de WebServices en software libre

- Complejidad en el desarrollo al momento de integrar componentes se verán expuestos así como sus métodos y datos correspondientes.
- Axis es una herramienta la cual requiere de varios conocimiento de sus APIS para el desarrollo y modificación de los métodos a exponer.
- Para este servidor JBoss la documentación tiende a ser una paradoja ya que gran parte de la documentación mas especifica requiere de una pequeña tarifa.
- No existe un soporte técnico utilizando software libre que garantiza la funcionalidad del software utilizado para las empresas
- Existe la posibilidad de encontrar problemas en cuanto a la capacidad y cantidad de datos que puede manipular en cierto tiempo
- Técnicamente, los datos a recibir y enviar carecen de una forma sencilla de manipular, es decir, no es fácil enviar una cantidad de datos a través de SOAP .
- Mejorar el protocolo SOAP que permite un mayor ancho de banda para la información.

CONCLUSIONES

La adaptación de componentes de distintas plataformas es un tema que aún está abierto, debido principalmente al creciente impulso que tienen las nuevas tecnologías, las cuales evolucionan rápidamente sin lugar a la elaboración de estándares de interconexión. En este sentido, la incorporación de elementos intermedios, como la utilización del lenguaje XML, facilita la interoperabilidad entre sistemas de software basados en plataformas heterogéneas, pensando como base para la creación de un protocolo donde se puede manejar datos en un solo estilo sin la complejidad de utilizar diversas plataformas o arquitecturas. Sin embargo, la inclusión de esta nueva capa de comunicación entre sistemas heterogéneos presenta ciertas características que deben se deben mejorar, incrementar el tiempo de ejecución de la comunicación vía XML ya que los participantes en la comunicación deben empaquetar/desempaquetar los datos de la comunicación en formato texto (XML). Además, han llegado a presentar atención en la seguridad que brinda la tecnología, ejecución de transacciones, viabilidad en el negocio electrónico, rendimiento de comunicación y futuro desde un punto de vista tecnológico aplicando al negocio y satisfacer necesidades de la empresa.

Es importante señalar que los estándares permiten facilitar los procesos de comunicación y satisfacer los procesos de desarrollo y de trabajo de las empresas, así como la opción de adquirir una infraestructura de hardware y software de acuerdo a sus necesidades y la oportunidad de operar con estándares apoyados, reconocidos por empresas líderes en el ámbito de las tecnologías de información.

WebService permite una comunicación basada en XML lo cual es importante hoy en día cuando se requiere unificar las aplicaciones y los sistemas, es decir, la idea es establecer y trabajar con estándares, los cuales sean debidamente conocidos, establecidos `para lograr la distribución de sistemas y aplicaciones. , tanto para SOAP, WSDL y UDDI entre otros

La utilización de WebServices como mecanismo para ofrecer servicios a plataformas heterogéneas, es uno de los mayores avances en cuanto a la interoperabilidad de sistemas software, estos contemplan ya los protocolos adecuados para la localización de servicios, ejecución remota de los mismos, empaquetado y desempaquetamiento de los datos.

En otro sentido, durante el despliegue de los sistemas software basados en componentes se debe garantizar que la interconexión entre los componentes se realiza entre partes previamente autenticadas, las cuales puedan demostrar que son quienes dicen ser, y hacen lo que dicen que hacen. Para ello, la incorporación de documentación adecuada a los componentes, unidos a un protocolo fiable de comunicación inicial entre los mismos debe garantizar la interconexión.

Actualmente, esta cuestión es motivo de debate en la comunidad científica, encontrándose razonamientos tanto a favor como en contra. Por un lado, la lógica de negocio representa una de las partes más críticas del sistema y los desarrolladores normalmente tienden a usar técnicas y herramientas estándares ampliamente usadas (en tiempo y espacio).

Tanto el desarrollo de software basado en componentes, como la utilización de WebServices cada día están más extendidas. Ello nos hace intuir un amplio futuro a ambas tecnologías, las cuales deberán aprender a coexistir, y serán utilizadas principalmente en aquellos sectores donde con más ventajas puedan ser explotados. En este sentido, dada la robustez de las plataformas de componentes serán estos los que formen la capa de modelo de negocio, dejando a los WebServices para formar la capa de presentación, los cuales facilitan la interconexión entre distintas plataformas y ofreciendo una interfaz común para el intercambio de información.

La propuesta de un desarrollo con software libre es descubrir la gama de posibilidades que se presentan hoy en día para el alcance de las empresas tanto en el aspecto económico, donde coadyuve a reducir y minimizar costos en cuanto a desarrollo de aplicaciones, adquisición de licencias de software, cambios posibles de ciertas plataformas que se requieren para conseguir los objetivos deseados.. Mejorar los procesos de producción de la empresa y de sus negocios; reducir el tiempo de transacciones y que estas sean efectivas y en tiempo real. Esto se traduce en que se debe considerar que la empresa que hace uso de la implementación de un Webservice enfocada a la aplicación, presenta una relación con el negocio electrónico que envuelve sus

sistemas de información: publicación, descubrimiento, localización, interoperabilidad, estándares, son características que buscan empresas líderes que apoyan el proyecto, puesto que ven en el un futuro de promover servicios los cuales requieren establecer un medio alterno y estrecho entre los sistemas que están empleando, lo anterior, basado en un estudio previo a su situación actual, el comportamiento de sus aplicaciones y analizara adecuadamente en que aspectos requiere el uso de un WebService, cuales métodos o datos requieren ser publicados para obtener un beneficio mutuo por parte de las empresas.

Por tanto, las opciones de desarrollo han sido hoy en día propuesta por plataformas como son J2EE y .Net, la idea es saber que existen posibilidades de optar por la alternativa que más convenga a la empresa y que no afecte principalmente sus costos de un proyecto de esta índole.

La capacidad que ofrece utilizar herramientas de software libre es que las empresas pueden pagar algunos desarrolladores, si es el caso, para adecuar y adaptar el software utilizado a sus necesidades y logrando una estabilidad en sus aplicaciones. Son el caso del uso de Linux, el de bases de datos, servidores de aplicación como JBoss que esta entrando en un mundo empresarial y el conjunto de sus distribuciones cada vez más sólidas y enfocadas a ser una opción más para los sistemas de información de las compañías.

Tanto el uso del servidor de aplicaciones JBoss que ha crecido y sigue creciendo en los últimos tiempo consolidándose como uno de los servidores de fuente abierta que contiene características similares a un servidor tipo comercial, principalmente características aun más apegadas al desarrollo con base a J2EE. Sus versiones adoptan módulos de seguridad , módulos WebServices, base datos, entre otra serie de servicios y módulos . donde su configuración y puesta en marcha resulta sencilla, así como su consola de administración JMX(Java Managment eXtensions) donde muestra los servicios activos y sus características respectivas. Lo mas importante es que pueden conseguirse los mismos resultados utilizando un servidor como JBoss y uno como WebSphere todo depende de la capacidad del sistema, es decir, el tamaño y los módulos que emplea la aplicación.

Para justificar el comentario anterior JBoss cumple su objetivo principal al momento de poder implementar varios servicios y módulos que hacen posible implementar cualquier tipo de aplicación basada en J2EE y a su vez estar en contacto con bases de datos tanto de tipo comercial y de licencia libre, es decir, los desarrolladores realmente se encuentra comprometidos establecer día con día versiones que mejorarán cada uno de sus módulos a través de un enriquecimiento por parte de la comunidad de licencia libre

La portabilidad es importante hoy en día al momento para la empresa y usuario s particulares. JBoss funciona adecuadamente en cualquier plataforma, puesto que existen versiones diseñadas para la mayoría de los sistemas operativos tanto en Windows y versiones en UNÍX. Lo cual representa una ventaja adicional al momento de escoger una herramienta que es versátil.

Así también, se puede apreciar que JBoss es capaz de adaptarse dentro de otras herramientas de software, lo cual facilita el proceso de desarrollo y administración. En un principio fue posible instalar conjuntamente el uso de un IDE como Eclipse capaz de facilitar el proceso de instalación, depuración, compilación y despliegue del servidor y, dentro de éste, la aplicación misma; pero el proyecto hizo uso de Apache Ant, la dificultad es que los desarrolladores antes de ello deben tener un conocimiento previo en los comandos y la estructura que guarda el archivo build.xml el cual permite definir el conjunto de comando y declaraciones de rutas de acceso, así como directivas de clases que facilitan los procesos de creación de éste proyecto, en otras palabras, las tareas mas comunes en una aplicación pueden ser compilación, depuración, despliegue, llenado de base de datos por mencionar uno ejemplos.

Pero la diferencia radica en que el software comercial provee kits de desarrollo llamados frameworks , que son simplemente entornos gráficos para implementar el WebService y el manejo de la información, pero la solución que recurre una empresa es por medio de una herramienta IDE como Eclipse que funciona tanto para servidores con software libre como comerciales principalmente WebSphere, Bea WebLogic entre otros.

Para el caso de Axis que es la base de un WebService contiene el conjunto de librerías capaces de implementar el protocolo SOAP y adaptarlo a las aplicaciones o módulos a exponer. Antes de ello se

considera algunos conocimientos previos del lenguaje Java, J2EE, el API correspondiente, conocer perfectamente como trabaja la aplicación, conocimiento de XML y HTML, entre otros elemento para al uso de Axis.
Así que para adoptar y realizar un WebService a través de Axis como software libre resulta de una tarea difícil para quien no tiene dichos conocimientos, y la empresas debe confiar en el equipo de desarrollo en el conocimiento de esto. Con lo anterior, se puede expresar que la viabilidad de un WebService con Axis brinda un soporte sólido y completo pero esto implica tener un conocimiento y un estudio total de la aplicación para construir o adaptar componentes. Como el caso de los ejemplos vistos en el proyecto, es un trabajo que muchas veces implica esfuerzo, ya que se puede recurrir a herramientas que simplifiquen tareas relacionadas a la construcción de un WebService en comparación a WebSphere. En contraste, los programas clientes tienen una ventaja de comunicarse con otra aplicación solo conociendo cirtos tipos de datos y modulos que estan expuestos en la interfas como es WSDL, para poder invocar un servicio o bien una serie de servicios, que pueden ser descubritos dentro de un regitro como lo es UDDI.

Los resultados de utilizar software Libre en la implementación de un WebService, demuestra la posibilidad de llevar a cabo dicha tarea , puesto que JBoss Axis , Ant son herramientas que hacen posible establecer la tecnología aplicada, mas lo interesante es poder conocer perfectamente los componentes de la aplicación a implementar , conocer cuales de ello puede ser publicados e invocados como WebServices si es un entitty bean, un servlet o simplemente una clase en particular hablando en componentes JAVA, así también , definir los tipos de datos con que se va interactuar y si es posible manipularlos, es decir, realizar el mapeo de objetos a texto, técnicamente hablando.

El software con licencia publica esta a la disposición, es importante conocer e investigar con profundidad la gama de opciones que presenta un código abierto, que también ha sido importante no solo por cuestión de reducir costos en las licencias, sino la oportunidad de modificar y aportar código y adaptarlo perfectamente a las necesidades de la aplicación o sistema con que se trabaje.

Vemos que es una opción competitiva en la cual las empresas en cierto momento deben considerar si los resultados y el desarrollo resulta competitivo en comparación con software comercial. Lo interesante es obtener del servidor resultados como escalabilidad en los servicios y módulos, soporte para la aplicación y sus demás componentes, seguridad,..

Se comprueba con base al proyecto establecido que es posible implementar un WebService con herramientas de licencia libre, pero intervienen ventajas como desventajas que pueden considerase dependiendo del ambiente y el entorno que se quiera utilizar.

Es factible el uso de WebService con Sofware Libre siempre y cuando se haya de establecer los benéficos y las ventajas que proporciona un escenario de esta naturaleza dentro de la empresa y obviamente en la aplicación misma, analizar si el contexto del problema o el entorno tanto los requerimientos de software, mantenimiento, seguridad, pruebas, desarrollo, etc.

Lo importante es mirar a que no requiera cambiar de plataformas ni adquirir software adicional ya que se ha de hablar de una comunicación basada en un estándar. El cual aun sigue en una etapa de evolución e investigación, quizás hoy en día requiere de mayor madurez, la cual puede presentarse y brindarla enorme cantidad de beneficios principalmente la formar de hacer negocio y brindar alternativas de lograr la inter-operabilidad de los sistemas. Hablado específicamente de las herramientas que proporciona el grupo de Fundación Apache y por parte de JBoss y la cierta aportación donde interviene empresas como es IBM y Sun Mycrosystems a este tipo de proyectos.

Uno de los problemas con los que se encuentra Internet es el de la integración y colaboración entre aplicaciones. Por tanto, se ve en los WebServices como un futuro en el cual se constituye como una alternativa de solución para lograr la integración de sistemas de computo siendo importante el uso de la estandarización. como el patron a utilizar el envio y recepción de datos.

La finalidad de la red es prestar servicio a las comunidades, los accionistas, los clientes y los empleados. Cada vez con mayor frecuencia, estos servicios se prestan a las comunidades a través de funciones de software que se han migrado a la red.

Este software puede estar distribuido, guardado por partes en los directorios de la red, o ser solicitado para realizar funciones específicas. Puede incluir aplicaciones Web, pero también puede contener diferentes innovaciones en aspectos como servicios Web u otros modelos de software por componentes.

Con los Web Services se puede reutilizar desarrollos ya utilizados sin importar la plataforma en la que funcionan o el lenguaje en el que están escritos. Los Web Services se constituyen en una capa adicional a estas aplicaciones de tal forma que pueden interactuar entre ellas usando para comunicarse tecnologías estándares que han sido desarrollados en el contexto de Internet.

Por otro lado, la finalidad de un WebService debe basarse en una tecnología aplicada al campo de los negocios electrónicos y lograr una funcionalidad y un objetivo de manipular estándares dentro de los sistemas computaciones y de las misma plataformas. Todo depende de la situación que se requiera resolver, analizar, diseñar y plantear si un WebService permite a las empresas resolver sus necesidades de optimizar y mejorar sus procesos; también, si sus aplicaciones, sus componentes o sus datos son propicios y adecuados para ser publicados y distribuidos. Por último, crear un plan a futuro donde el WebService no quede como una aplicación obsoleta y no rinda los frutos esperado. Por ello como todo proyecto esto conlleva a una seria pasos con la finalidad de planear la situación de la empresa, el objetivo que persiguen con una adaptación tecnológica diferente capaz de brindar mejores resultados en las actividades de las empresas, principalmente aplicado a la publicación y descubrimiento de productos y servicios a través de la página WSDL ubicada en UDDI.
Así el contacto entre sistemas de varias empresas puede ser motivo de mejorar sus negocios y evitando pérdidas de tiempo en encontrar proveedores en línea y que necesitan información actualizada de forma inmediata, segura y coherente.
Cabe señalar, que los WebServices tienen un desarrollo muy reciente y como toda tecnología resulta de un trabajo de investigación aplicado a la creación de diversos mecanismos, software, etc. y sobretodo de un apoyo en el funcionamiento y características de SOAP. La idea de tener contactos con aplicaciones de diversos sistemas resulta atractiva, siempre y cuando la empresa considere su entorno, sus objetivos de promoción, conocer otras alternativas diferentes a WebServices y comprobar si realmente es indispensable el uso de una tecnología como esta.

Finalmente, Los Web Services representan una posibilidad para que dos o más programas se comuniquen entre sí sin tener que mediar intervención humana. Se ha utilizado el término Web Services para cubrir lo que se conoce como servicios de aplicaciones provistos por la Web, como el acceso a una agenda de un grupo de trabajo, sin embargo el verdadero impacto que tendrán las tecnologías se sentirá cuando se empiecen a usar profusamente los estándares en las conexiones entre aplicaciones en modelos de comercio negocio a negocio. En últimas lo que se quiere es que cualquiera que desee uno de nuestros servicios pueda acceder a un código de programación y colocarlo en su sitio Web, que con la suficiente seguridad en el proceso, podríamos estar efectuando una extensión del sistema a terceros con la sencillez que provee el uso de estándares.

Con el advenimiento de la tendencia de los WebServices y las tecnologías que nos aporta, es factible ahora si efectuar programación en cada uno de los interesados en conectarse sin preocuparse por la marca de la tecnología, siempre y cuando observe los estándares. A manera de ejemplo, un sitio de alquiler de vehículos puede querer que en todos los sitios Web de las aerolíneas se pueda alquilar un vehículo y a la aerolínea le interesará que la persona no tenga que salirse del sitio Web de la aerolínea para efectuar la transacción. Si la empresa que arrienda vehículos tuviera que hacer un programa para cada aerolínea, estaría en dificultades, sin embargo con los WebServices y sus tecnologías, con un solo desarrollo estaría disponible para todas las aerolíneas que se quieran incorporar al plan.

Sin estándares ampliamente aceptados por todos, para lo que se requiere que sean independientes del proveedor, los WebServices pueden terminar sin tener frutos acompañando a otros intentos para unificar la lógica de los negocios, como Corba, DCOM, EDI, Java Remote Method Invocation y UNIX Remote

Procedure Call. Solo WebServices provee un desarrollo y futuro a largo plazo, debido a que la característica principal es el manejo de estándares, es decir, un solo estilo o forma de enviar y recibir datos entre aplicaciones.

Primero las empresas cuidan de sus recursos informáticos y tecnológico, pero también es importante señalar que aun existe software comercial que provee mayor confiablilidad en cuanto a soporte técnico, documentación, seguridad, garantía en hardware y software. Esto permite a las compañías no arriesgar su negocio y prefieren utilizar herramientas a pesar de pagar costosas licencias. Al parecer, esto es un punto normal en el cual depende simplemente de la situación y nivel de una organización.

Los WebServices pueden ser muy útiles en ciertos casos concretos de programación. Los expertos de marketing de empresas de software como Microsoft anuncian una revolución debido a la aparición de SOAP, WSDL, UDDI, etc. Sin embargo, es una tecnología que sigue creciendo y la solución a primera vista es proveer comunicación entre aplicaciones, lo que es Internet para los humanos.

La única revolución que ha de vivir en relación a los WebServices es la de ver como los servidores de Internet hablan entre ellos, y cada día son más independientes del programador. Y eso es algo que va a pasar completamente desapercibido para el resto de los usuarios de Internet.

Si se logra que existan WebServices de utilidad, gratuitos y sencillos, este nuevo esquema de comunicación y programación tendrá futuro. Si por el contrario con la finalidad de entrar en el negocio se tiene la idea de un WebService como una tecnología de moda sin conocer los propósitos esenciales de éste y el futuro de nuevos protocolos de comunicación, se perderá un proyecto que resulte de gran impacto evolución tecnológica aplicada y enfocada al negocio.

Actualmente las empresas con los cambios en la economía, la globalización, y otros factores que modifican el ambiente competitivo, buscan alternativas de solución, y las identifican como estrategias para su negocio.

Hoy en día, las TI juegan un papel importante en este ambiente, ya que facilitan a las empresas en una forma más rápida, eficiente y efectiva la presentación de sus servicios a través del uso de Internet y de forma estar a la vanguardia y así mismo a distribuir la información generada a lo largo de la empresa.

Las empresas que han identificado WebServices como estrategias de negocio, se han dado a la tarea de enfocarse en desarrollar sus propias metodología de planeación estratégica de WebServices, para ajustarlas a las necesidades a las que están expuestas, y así satisfacerlas y ofrecer mejores servicios a sus clientes. Otorgándoles un valor agregado y métricas justificables de inversión.
Es por esto, que la planeación estratégica de un WeBServices es una herramienta útil que permite a las empresas ajustarse a estos cambios, donde la planeación estratégica, debe ser un proceso que les permita alinear sus objetivos, misión, visión y procesos de negocio con las estrategias definidas.

Intuitivamente un Web Serviré 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 WebService, 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.

Empresas líderes están promocionando SOAP como estándar de los mensajes para los Web Services. Un mensaje SOAP se parece mucho a una carta: es un sobre que contiene una cabecera con la dirección del receptor del mensaje, un conjunto de opciones de entrega (tal como la información de encriptación), y un cuerpo o body con la información o data del mensaje.

Promocionan los WebServices como un modelo de programación para la comunicación entre aplicaciones. Estas compañías piensan que la conexión de aplicaciones a través de la Internet mejorará la capacidad de las empresas para trabajar conjuntamente con sus socios de negocio, proveedores y clientes. Creando una capa de

Web Services sobre una aplicación corporativa existente, las organizaciones podrán permitir que sistemas externos puedan invocar las funciones de la aplicación a través de Internet (o una intranet corporativa) sin tener que modificar la aplicación misma. Por ejemplo, varias compañías están hoy en día creando Web Services que actúan como front end para aplicaciones de entrada de órdenes que están residentes internamente en un mainframe. Estas compañías permiten a los sistemas de compras de sus clientes enviar órdenes de compra a través de la Internet. Poner una capa de WebServices sobre las aplicaciones existentes es una solución muy interesante para integrar las aplicaciones desarrolladas por los diferentes departamentos y así reducir los costos de la integración

El mundo de los Web Services está evolucionando a gran velocidad. En los últimos años, las especificaciones de los Web Services han sido definidas, refinadas y a veces, criticadas; se han lanzado kits de herramientas y los desarrolladores los han utilizado para construir sistemas, y los fabricantes han trabajado para garantizar la interoperabilidad al igual que con la infraestructura Web clásica, se está desarrollando e implantando tecnologías en paralelo.

Actualmente, la plataforma de WebServices ha sido desarrollada lo suficiente para crear aplicaciones distribuidas que se comunican enviando mensajes SOAP sobre HTTP. Esto no significa que no existan herramientas o marcos de trabajo de Web Services que puedan hacer más cosas, por ejemplo, enviar mensajes SOAP sobre SMTP. Sin embargo, la mensajería basada en HTTP es la única opción de comunicación más generalmente soportada. La mayoría de las herramientas de Web Services mapea mensajes SOAP a invocaciones de métodos en objetos. Algunas proporcionan además un acceso directo al cuerpo de los mensajes SOAP, exponiéndolos como XML

La mayoría de los kits de herramientas de Web Services tal es como Axis interoperan bastante bien, especialmente si limita a formatos de mensajes relativamente sencillos. Los miembros de la comunidad SOAPBuilders han realizado grandes mejoras sobre interoperabilidad organizando evaluaciones de SOAP y WSDL, y reuniéndose periódicamente para que los kits de herramientas trabajen periódicamente.

Muchos de los trabajos de la comunidad SOAPBuilders han sido necesarios en relación con varios aspectos de las especificaciones SOAP 1.1 y WSDL 1.1. El W3C dispone de grupos de trabajo localizados en refinar ambas especificaciones, lo que debería aportar mejoras sustanciales. El grupo XML Protocol Working Group está trabajando en la especificación SOAP 1.2 mientras que el grupo Web Services Description Working Group está creando la especificación WSDL 1.2.

Mientras que el trabajo en el W3C se centra en las nuevas versiones de las especificaciones del núcleo de los Web Services, existe otra organización independiente que está prestando más atención a la interoperabilidad. La organización Web Services Interoperability Organization (WS-I) tiene como objetivo definir mejores prácticas para garantizar la interoperabilidad de los servicios Web. El grupo WS-I Basic Profile Working Group está actualmente desarrollando un conjunto de recomendaciones sobre cómo utilizar los protocolos básicos de los servicios Web como SOAP 1.1 y WSDL 1.1 para maximizar la interoperabilidad.

El trabajo sobre la plataforma de Web Services continuará en el futuro, y aparecerán mejoras en tres áreas fundamentales. En primer lugar, se añadirán servicios de más alto nivel. Todo el mundo está de acuerdo en que debe existir un modo estándar de asegurar Web Services, rutear mensajes, garantizar una entrega fiable de mensajes, definir semántica transaccional, etc. Estas características de propósito general se expanden más allá de los dominios del problema y no hay ninguna razón por la que cada desarrollador de Web Services deba implementarlas individualmente. Microsoft, IBM y otros están realizando mucho trabajo en estas áreas.

En segundo lugar, seguirán estandarizándose especificaciones. El ciclo de vida de las especificaciones de Web Services típicamente progresa desde una propuesta hasta un estándar de facto y desde éste hasta un estándar real. Con SOAP 1.2 y WSDL 1.2 las peculiaridades finales están siendo elaboradas de las especificaciones SOAP y WSDL y algunas de las especificaciones de servicios de mayor nivel, como WS-Security, ya están en el proceso de estandarización. Las empresas siguen proponiendo nuevas especificaciones como añadidos a la plataforma de Web Service y la industria en su conjunto necesitará acordar cuáles adoptar. Esas especificaciones necesitarán a continuación ser estandarizadas.

En tercer lugar, los kits de herramientas y marcos de trabajo seguirán mejorando. Además de servicios de más alto nivel como la seguridad y los objetos adjuntos, se añadirá soporte para protocolos de transporte alternativos como SMTP o TCP. De modo más importante, los modelos de programación migrarán desde los servicios de tipo RPC hacia servicios centrados en documentos, para promover un acoplamiento débil.

Llegados a este punto, la cuestión es cómo utilizar los Web Services cuando la plataforma todavía está evolucionando. Las herramientas y marcos de trabajo de los Web Services actuales proporcionan suficiente funcionalidad básica para desarrollar interesantes aplicaciones distribuidas que envían mensajes SOAP sobre HTTP. Algunos servicios de mayor nivel como WS-Security están empezando a refinarse con el soporte de diversas herramientas nuevas como el kit Web Services Development Kit. Otros servicios están todavía en fase de desarrollo preliminar a medida que las especificaciones se van revisando y las primeras implementaciones exponen áreas en las que las especificaciones necesitan solidificarse. Mientras tanto, podemos apoyarnos en mecanismos HTTP tradicionales para implementar seguridad y demás características.
Los WebServices son un vehículo para mejorar la comunicación y el compartimento de datos de diversos sistemas entre agencias y él publico ,las tecnologías de información buscan extenderse mas allá de las redes de la organización siempre y cuando buscando una nueva opción tecnológica , quizás visto como un esquema diferente lo que proporciona XML a través de los protocolos como SOAP, WSDL y UDDI.

Resumiendo:

- WebServices basados en el lenguaje XML, por el cual se ha creado un protocolo de comunicación o transmisión de datos a través de aplicaciones de diversas plataformas: SOAP, WSDL, UDDI
- WebServices es una tecnología de aplicación a los negocios electrónicos específicamente a la publicación e identificación y descubrimientos de servicios donde la intefaz queda a cargo de WSDL .
- Tiene el soporte de desarrollo por diversas corporativos tal es IBM, Microsoft, Hewlett Packard, Sun Microsystems, etc. Los cuales siguen aportando innovaciones tecnológicas integrándolos en sus productos y desarrollando herramientas para incrementar su soporte correspondiente.
- WebServices ha tenido el soporte alternativo de la comunidad de software libre, entrando a través de los servidores de aplicación que han integrado módulos capaces de contener WebServices y el conjunto de herramientas relacionados a éste, así como también un sistema operativo como lo es Linux
- Hablar de la factibilidad de un WebServices a través de software libre ha sido estudiado bajo un esquema del negocio electrónico, enumerando las ventajas y desventajas que intervienen en el desarrollo de los procesos de la empresa. Así también, basarse en la productividad y funcionamiento al exponer módulos de una aplicación en particular donde las empresas dará a conocer servicios y/o productos
- Linux, JBoss Axis son los componentes de software libre que han demostrado la capacidad de sostener una aplicación basada en J2EE y lograr que los módulos de ésta última, en particular el conjunto de EJB, logren ser expuestos como WebServices.
- Los resultados obtenidos demuestran que el funcionamiento de un WebService es posible, demostrando que a través de una arquitectura totalmente opuesta como lo es un cliente Visual Studio .Net puede interactuar con una aplicación J2EE.
- La problemática que aun existe es poder tener una capacidad de manipulación de los datos a través del protocolo SOAP, incrementar y aportar mejoras por parte de las empresas que soportan el proyecto. Sin embargo, existen estrategias capaces de brindar la seguridad por medio de métodos conocidos como son encriptación, firewalls, etc..
- Al adquirir software libre y el conjunto de herramientas es un beneficio la reducción de costos al momento de no recurrir al pago de una licencia. Pero, la empresa debe poner su confianza en un grupo de desarrolladores en cuanto a soporte e innovación de productos. Con base a Linux y JBoss están incursionando aun más en el campo empresarial, estudios realizados basan la competitividad de estos mismo logrando resultados satisfactorios para las empresas, quizá el aspecto de difusión y la cultura de la empresa es no arriesga a migrar sus sistemas por falta de información, y prefieren la garantía y el soporte que brinda el pago de licencia y adquisición de software comercial.
- Por lo que se aprecia hoy en día la comunidad de software libre esta incursionando en mejorar y proveer mejores productos, soporte, información en línea, comunicados de versiones y actualizaciones,

conferencias, etc. acerca de productos competitivos y enfocados también a proveer herramientas adecuadas y adaptables a satisfacer los beneficios de las empresas.

- Por último en cuestión de software libre países han pensado migrar sus sistemas a Linux y con ello el conjunto de programas asociados a éste, con este dato se puede pensar un desarrollo tecnológico de cualquier índole, en este caso WebService. Brindar una alternativa económica y competitiva que puede ser adaptable siempre y cuando se tenga un conocimiento de las aplicaciones a manejar, el objetivo de la empresas, su situación actual en cuestión financiera y de adquisición tecnológica, y analizar si la compañía o negocio electrónico se satisface de una comunicación con diversas aplicaciones tanto de clientes como de proveedores que requieren obtener dato de forma eficaz , confiable e inmediata.
- El funcionamiento que provee un sistema operativo como Linux y un servidor de aplicación J2EE JBoss que pude ser instalado en diversas plataformas , ya que provee la estabilidad de la aplicación, las pruebas de la aplicación Dukes Bank ha sido satisfactorio, la aplicación no requiere de muchas modificaciones para correr en el servidor de aplicación, JBoss y sus nuevas versiones demuestran que sus módulos dan el soporte y herramientas basadas en servidores J2EE, un mejor desempeño.
- Así también lo importante es la seguridad de la aplicación el cual JBoss contiene módulos de seguridad como JAAS(un módulo de seguridad de autenticación para J2EE), entre otras opciones, así que la aplicación Dukes Bank toma y basa su seguridad en el mismo servidor de aplicación.
- JBoss como multiplataforma brinda una opción para las empresas de concentrar sus aplicaciones y módulos en WebServices.
- Finalmente la empresa tiene un beneficio a través de WebServices de promover servicios en Internet a través de UDDI y tener una relación más estrecha con el cliente, todo depende del costo y del funcionamiento que desee adoptar.
- Todo este proyecto depende mucho de la situación actual de la empresa, de su cultura corporativa, cultura informática, de establecer un plan adecuado en sus sistemas de información. Es recomendable la tecnología Web Services, como cualquier otra, utilizarla sólo si es necesario y es capaz da adaptarlo al entorno y a la problemática o innovaciones a realizar en los procesos de las empresas, siempre y cuando ofrezca beneficios de promoción de negocio , en síntesis una inversión. Las opciones están al alcance tanto con software comercial, y de acuerdo a los resultados obtenidos en el presente trabajo, el software libre esta en un momento de oportunidad de competir y demostrar que se pueden lograr cosas interesantes teniendo la misma respuesta, funcionalidad sin arriesgar en la seguridad de sus sistemas de información.

ANEXO I

(a)

```
  POST /servlet/rpcrouter HTTP/1.1
  Host: www.messages.com
  Content-Type: text/xml; charset="utf-8"
  Content-Length: nnnn
  SOAPAction: ""

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  <SOAP-ENV:Body>
      <ns1:getMessage xmlns:ns1="urn:NextMessage"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <UserID xsi:type="xsd:string">JDoe</UserID>
        <Password xsi:type="xsd:string">0JD0E0</Password>
      </ns1:getMessage>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Mensaje SOAP embebido en una solicitud HTTP

(b)

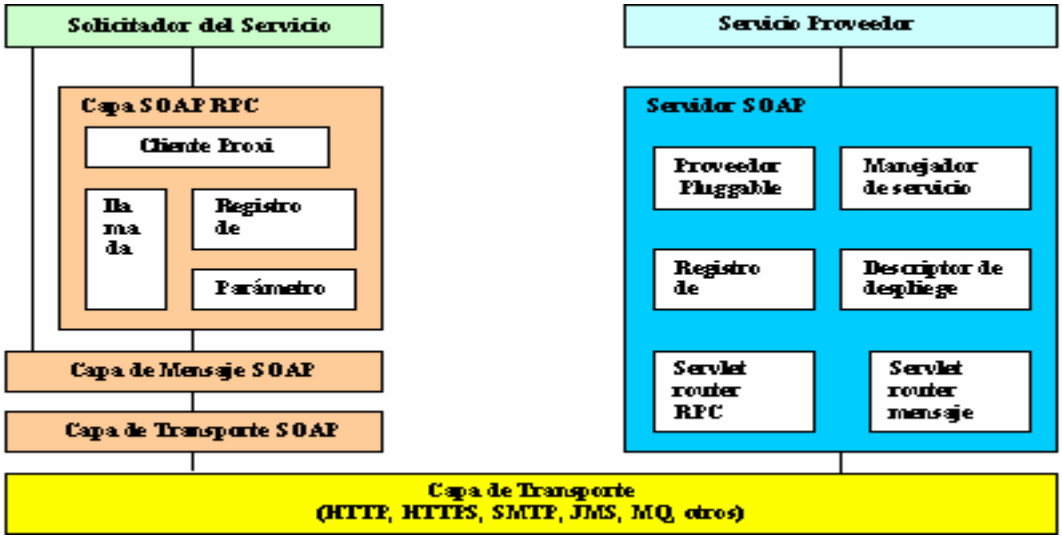
```
  HTTP/1.1 200 OK
  Content-Type: text/xml; charset="utf-8"
  Content-Length: nnnn

<SOAP-ENV:Envelope>
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  <SOAP-ENV:Body>
      <ns1:getMessage xmlns:ns1="urn:NextMessage"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <return xsi:type="xsd:string">Call mom!</return>
      </ns1:getMessage>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Mensaje SOAP embebido en una respuesta http

(c)

Se muestra la arquitectura conceptual del proveedor de servicio (Servidor SOAP) y el solicitador de servicio (cliente SOAP).



Modelo de componentes SOAP de alto-nivel

ANEXO II

CAPITULO II

(a) y (b)

EJEMPLO

Un simple documento WSDL contiene un buen numero de elementos con diferentes relaciones a cada uno. Figura (a) y Figure (b) contienen el archive WSDL de ejemplo. El cual seguirá siendo analizado después con más detalle.
El ejemplo es suministrado como un único archivo. Aunque es posible fragmentar el archivo WSDL en mas de una parte.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://address.jaxrpc.samples"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:apachesoap="http://xml.apache.org/xml-soap"
   xmlns:impl="http://address.jaxrpc.samples"
   xmlns:intf="http://address.jaxrpc.samples"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
  <schema targetNamespace="http://address.jaxrpc.samples"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="AddressBean">
    <sequence>
     <element name="street" nillable="true" type="xsd:string"/>
     <element name="zipcode" type="xsd:int"/>
    </sequence>
   </complexType>
   <element name="AddressBean" nillable="true" type="impl:AddressBean"/>
  </schema>
  <schema targetNamespace="http://www.w3.org/2001/XMLSchema"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <element name="int" type="xsd:int"/>
   <element name="string" type="xsd:string"/>
  </schema>
 </wsdl:types>
   <wsdl:message name="updateAddressRequest">
      <wsdl:part name="in0" type="intf:AddressBean"/>
      <wsdl:part name="in1" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="updateAddressResponse">
      <wsdl:part name="return" type="xsd:string"/>
   </wsdl:message>
   <wsdl:portType name="AddressService">
      <wsdl:operation name="updateAddress" parameterOrder="in0 in1">
         <wsdl:input message="intf:updateAddressRequest"
                     name="updateAddressRequest"/>
         <wsdl:output message="intf:updateAddressResponse"
                     name="updateAddressResponse"/>
      </wsdl:operation>
   </wsdl:portType>
```

(a) Ejemplo simple de un WSDL parte 1

```
   <wsdl:binding name="AddressSoapBinding" type="intf:AddressService">
      <wsdlsoap:binding style="rpc"
                        transport='http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="updateAddress">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name='updateAddressRequest">
            <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/'
                namespace='http://address.jaxrpc.samples" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="updateAddressResponse'>
            <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/'
                namespace='http://address.jaxrpc.samples" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="AddressServiceService">
      <wsdl:port binding="intf:AddressSoapBinding" name="Address">
         <wsdlsoap:address
                location="http://localhost:8080/axis/services/Address"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

(b) Ejemplo simple de un WSDL parte 2

(c)

WSDL USA LOS XML NAMESPACE (Nombres de Espacio) LISTADOS EN LA TABLA

Prefijo	Espacio de Nombres URI	Explicación
wsdl	http://schemas.xml.soap.org/wsdl/	Espacio de nombres para WSDL framework
soap	http://schemas.xml.soap.org/wsdl/soap	SOAP Binding
http	http://schemas.xml.soap.org/wsdl/http	http Binding
mime	http://schemas.xml.soap.org/wsdl/mime	MIME Binding
soapenc	http://schemas.xml.soap.org/soap/encoding	Nombres de espacio de Codificación
soapenv	http://schemas.xml.soap.org/soap/envelope	Nombres de espacio del Sobre
xsi	http://www.w3.org/2000/10/XMLSchema-instance	Nombre de espacio de instancia
xsd	http://www.w3.org/2000/10/XMLSchema	Nombres de espacio de esquema
tns	URL a el archivo WSDL	Usado por una convención para referirse a los documentos actuales

Los primeros 4 nombres de espacio son definidos por la misma especificación WSDL; las siguientes 4 definiciones referenciadas a nombres de espacios que son definidas en el SOAP y estándar XSD. La ultima es local para cada especificación, en el ejemplo no se usan reales espacios de nombres, la URI contiene localhost.

(d)

El siguiente ejemplo mostrado, es donde se especifica un tipo complejo llamado AddressBean, el cual es compuesto de dos elementos, una calle y una zona postal. Se especifica que el tipo de dato de la calle (street) es una cadena (string) y el tipo de dato de la zona postal (zcode) es un numero (int).

```
<wsdl:types>
  <schema targetNamespace='http://address.jaxrpc.samples'
            xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/'/>
   <complexType name="AddressBean'>
    <sequence>
     <element name='street' nillable='true' type='xsd:string"/>
     <element name='zipcode" type="xsd:int"/>
    </sequence>
   </complexType>
   <element name='AddressBean" nillable="true" type="impl:AddressBean"/>
  </schema>
  <schema targetNamespace='http://www.w3.org/2001/XMLSchema"
         xmlns="http://www.w3.org/2001/XMLSchema'>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/'/>
   <element name='int" type="xsd:int"/>
   <element name='string' type='xsd:string'/>
  </schema>
 </wsdl:types>
```

Tipo de definición en el documento WSDL

ANEXO III

Capitulo III

(a)

Registros UDDI

Organización Hosting	Tipo de Acceso	URL
Registro de Negocio IBM	Web	http://uddi.ibm.com
	Investigación	http://uddi.ibm.com/ubr/inquiryapi
	Publicar	https://uddi.ibm.com/ubr/publishapi
Registro de Prueba IBM	Web	http://uddi.ib,.com/testregistry/registry.html
	Investigación	http://www-3.ibm.com/services/uddi/testregistry/inquiryapi
	Publicar	https://uddi.ibm.com/testregistry/publishapi
Registro de Negocio Microsoft	Web	http://uddi.microsoft.com
	Investigación	http://uddi.microsoft.com/inquire
	Publicar	https://uddi.microsoft.com/publish
Registro de Prueba Microsoft	Web	http://test.uddi.microsoft.com
	Investigación	http://test.uddi.microsoft.com/inquire
	Publicar	https://test.uddi.microsoft.com/publish
Registro de Negocio SAP	Web	http://uddi.sap.com/
	Investigación	http://uddi.sap.com/uddi/api/inquiry
	Publicar	https://uddi.sap.com/uddi/api/publish
Registro de Prueba SAP	Web	http://udditest.sap.com/
	Investigación	http://udditest.sap.com/UDDI/api/inquiry
	Publicar	https://udditest.sap.com/UDDI/api/publish
Registro de Negocio NTT	Web	http://www.ntt.com/uddi
	Investigación	http://www.uddi.ne.jp/ubr/inquiryapi
	Publicar	https://www.uddi.ne.jp/uddi/ubr/publishapi

ANEXO IV La API base para Web Services en la plataforma Java (JAX-RPC)

La API Java para XML –basado en RPC (JAX-RPC) habilita a los desarrolladores de tecnología Java para el desarrollo de SOAP basado en la interoperabilidad y portabilidad de los Web Services. JAX-RPC provee el núcleo API de desarrollo y despliegue de Web Services en plataforma Java. JAX-RPC requiere parte de la plataforma J2EE 1.4.

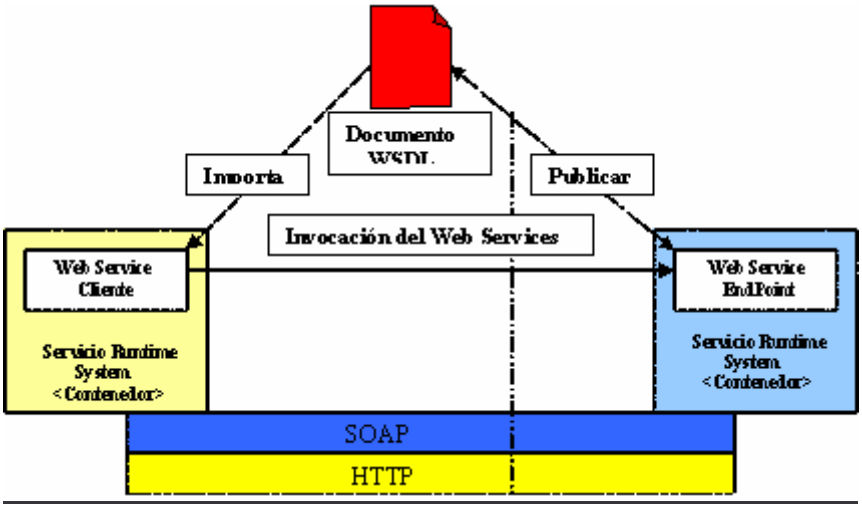
Caracteristicas y beneficios de desarollo

- Web Services Portables e Inter-Operables
- Facilidad de desarrollo de punto final y clientes Web Services
- Productividad creciente de desarrollo
- Apoyo para estándares abiertos: XML, SOAP. WSDL.
- Soporte para herramientas.
- Modelo de programación RPC
- Soporte para modelos de procesamiento de mensajes SOAP y extensiones.
- Seguridad en los Web Services
- Variado tipo de mapeo

Los desarrolladores utilizan el modelo de programación JAX-RPC para desarrollar Web Services endpoints y clientes basados en SOAP. Un endpoint se describe usando un documento Lenguaje de descripción Web Services (WSDL). JAX-RPC habilita clientes JAX-RPC para invocar Web Services desarrollados a través de plataformas heterogéneas. De forma similar, Web Services JAX-RPC requieren de los estándares SOAP y WSDL para la interoperabilidad de distintas plataformas y lenguajes. Esto hace a JAX-RPC una llave para la tecnología de los Web Services basados en la integración.

JAX-RPC provee soporte para un mapeo de WSDL->Java y Java ->WSDL como parte del desarrollo de los web services clientes y endpoints. En un entorno de desarrollo típico, las herramientas proveen funcionalidad a ese mapeo. Esto además simplifica el desarrollo de aplicaciones.

JAX-RPC habilita un web services endpoint para ser desarrollado para usarlo como componente Servlet o EJB(Enterprise Java Beans).Un web services endpoint se desarrolla en un contenedor web o un contenedor EJB basado en el modelo correspondiente. Esos endpoint se describen usando un documento WSDL. Este documento WSDL puede ser publicado en un registro público o un registro privado. Un cliente utiliza un documento WSDL e invoca el web services endpoint.



Estructura JAX-RPC

JAX-RPC requiere de SOAP sobre http para la interoperabilidad, JAX-RPC soporta modelos de procesamiento para mensajes SOAP. Esto permite a los desarrolladores construir extensiones específicas para el soporte en la seguridad, acceso y cualquier otra facilidad basada en los mensajes SOAP. JAX-RPC usa SAAJ API para mensajes de administradores. SAAJ provee una API de Java para construir y manipular mensajes SOAP con archivos. Esto significa que permite el intercambio de documentos XML, imágenes y cualquier oto tipo de archivo a través del Web Services.

JAX-RPC soporta administración de sesiones de nivel http y SSL basado en mecanismos de seguridad. Esto permite a los desarrolladores crear web services seguros.

JAX-RPC

Un RPC – basado en Web Services es una colección de procedimientos que pueden ser llamados por un cliente remoto sobre el Internet. Por ejemplo, un Web Service basado en RPC es un servicio de cotización de existencias que toma una petición SOAP (Simple Object Access Protocol) para el precio de un producto y regresa el precio vía SOAP.

Un Web Service, es un servidor de aplicación que implementa los procedimientos que están disponibles para que los clientes se comuniquen, se desarrolla en un contenedor o server-side. El contenedor puede ser un contenedor servlet tal como Tomcat o un contenedor en una Plataforma Web Java 2, Edición Empresarial (J2EE).

Un Web Services puede por si mismo hacer disponible clientes potenciales describiendo por si mismo con un Documento (WSDL). Un WSDL es un documento XML que da la información pertinente acerca de un Web Service, incluyendo su nombre, las operaciones que pueden ser realizadas en él, los parámetros para esas operaciones, y la localización de donde enviar las peticiones. Un consumidor (Cliente Web) puede usar el documento WSDL para descubrir lo que el servicio ofrece y como accesarlo.

Interoperabilidad

Con JAX-RPC, un cliente escrito en lenguaje diferente a Java puede acceder a un Web Service desarrollado y desplegado en la plataforma Java y viceversa. Esto quiere decir que un servicio puede desarrollarse y desplegarse usando otra plataforma.

Lo que hace esta interoperabilidad posible que el soporte de JAX-RPC es por SOAP t WSDL. SOAP define estándares para mensajes XML y el mapeo de tipo de datos así que la aplicación adherida a esos estándares que pueden comunicarse uno con otro. JAX-RPC es un procedimiento de comunicación remoto que se implementa en mensajes de petición y respuesta SOAP:

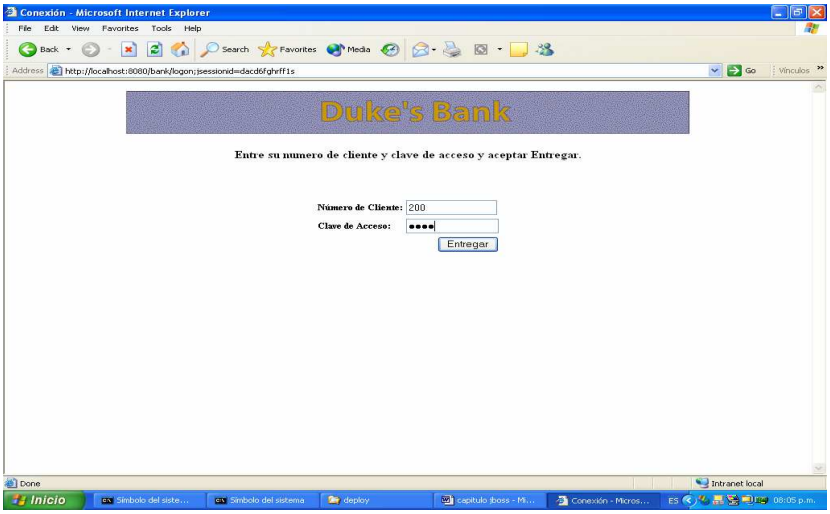
La otra clave de la interoperabilidad es que JAX-RPC soporta WSDL. EL documento WSDL hace la descripción del servicio portable.

Fácil uso

Dado que JAX-RPC esta basado en un mecanismo (RPC) de procedimientos remotos, su desarrollo es amigable aunque envuelve un serie de infraestructuras complicadas, pero JAX-RPC hace posible esconder los detalles al cliente y al desarrollador del servicio. Por ejemplo, un cliente de Servicio Web simplemente hace métodos de comunicación Java, y todos los detalles de la transmisión se realizan de forma automática. En el lado del servidor, el Web Services simplemente implementa el servicio que ofrece y, como el cliente, no le interesa saber los mecanismos de implementación.

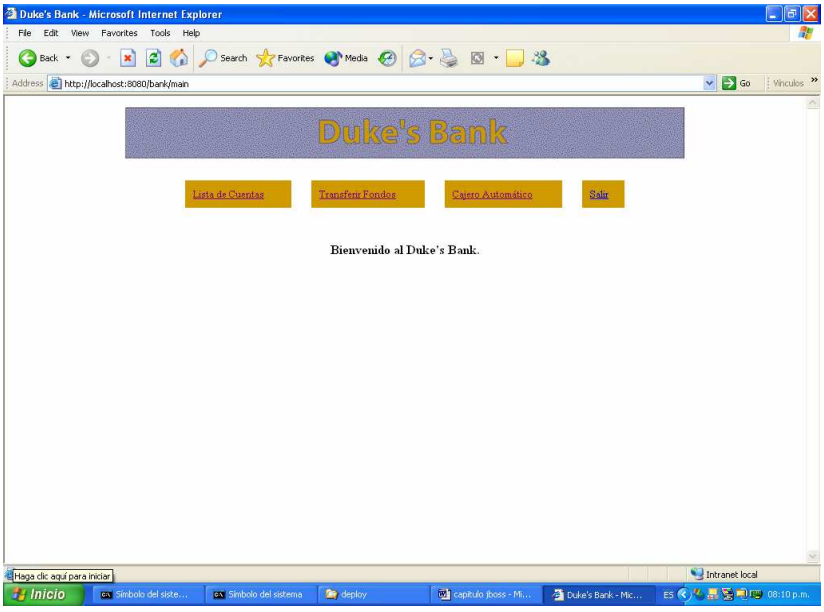
En conclusión debido a la facilidad de uso, JAX-RPC es la principal API para Web Service para aplicaciones cliente y servidor. JAX-RPC se concentra en mensajes SOAP punto - a - punto, el mecanismo básico que muchos clientes de Web Services usan.

ANEXO V FUNCIONAMINETO DE LA APLICACIÓN DUKESBANK

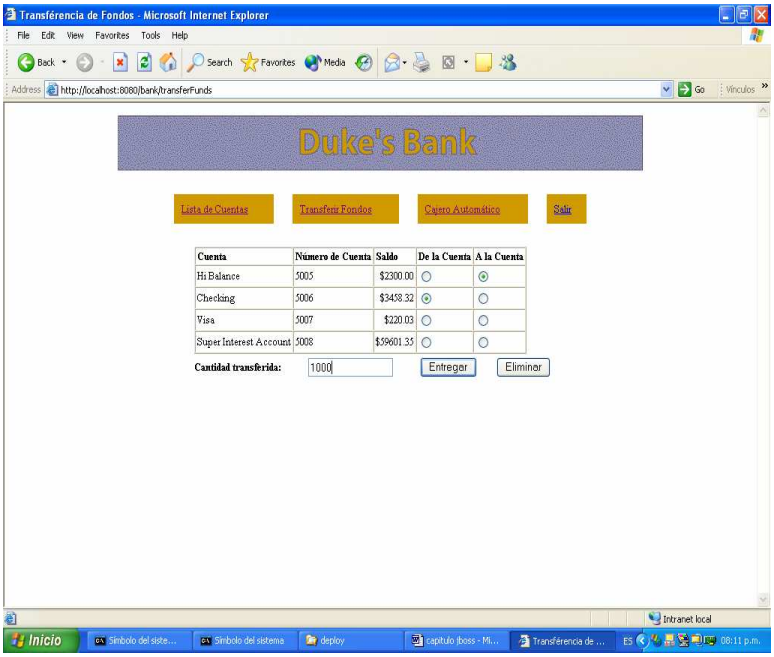


Página principal de la aplicación: Autenticación

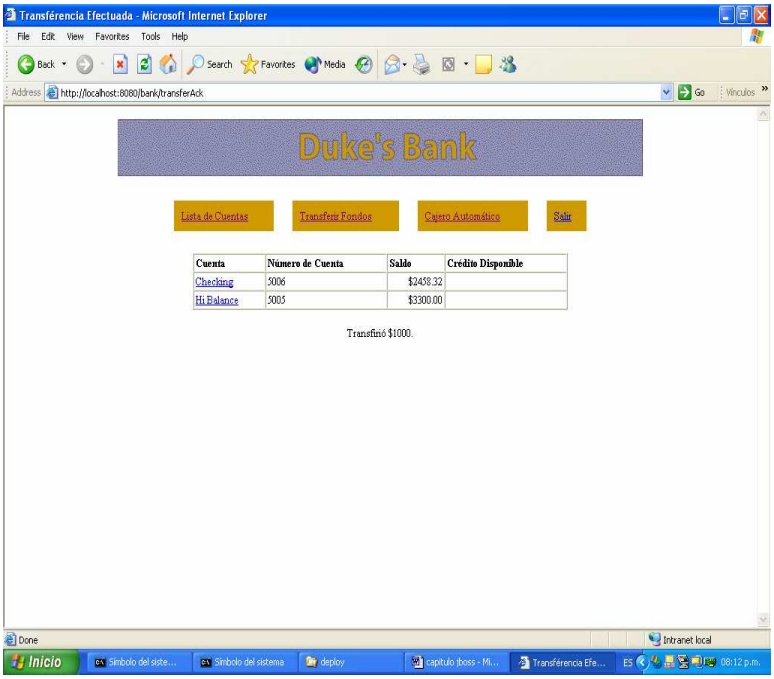
Dentro de la aplicación hay un conjunto de operaciones. Lista de cuentas, Trasferencia de Fondos, Cajero Automático y la opción de salida



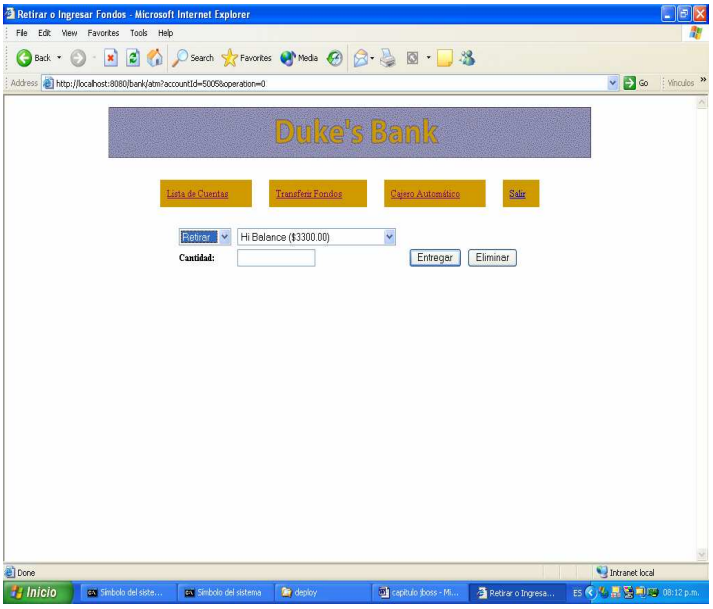
Menú de transacciones de la aplicación



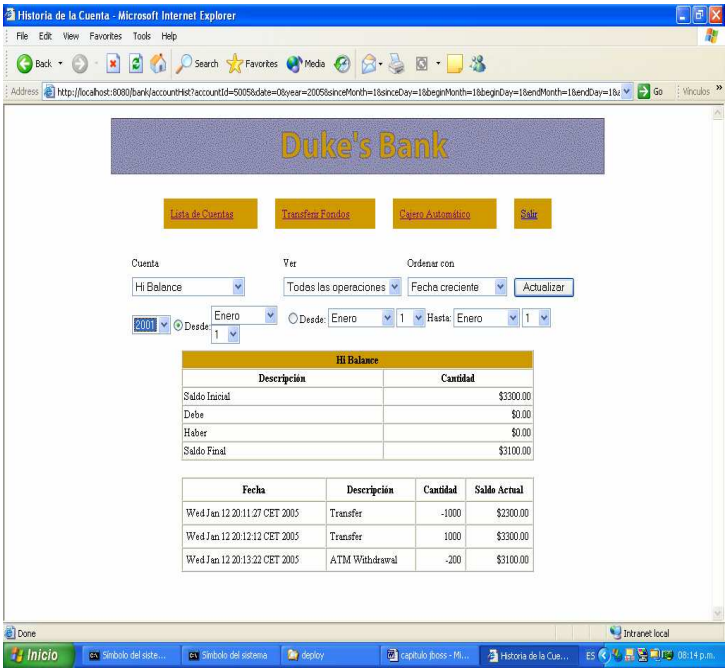
. Transferencia de cuentas



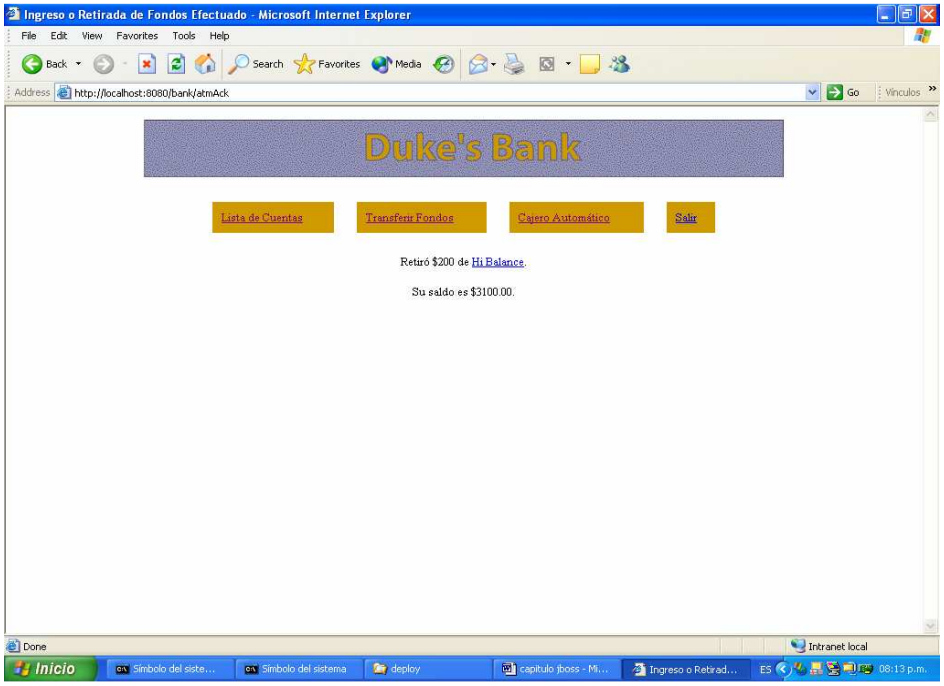
Saldo de cuentas



Cajero automático ATM



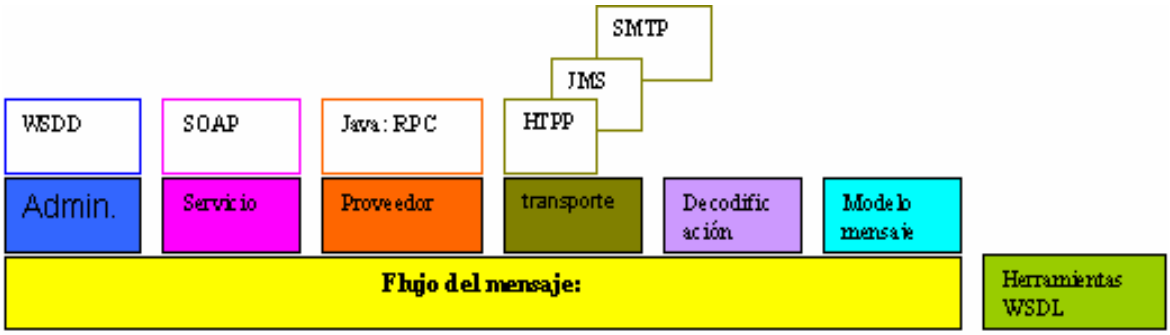
Estado de Cuenta



Resultado de transacción

ANEXO VI SUBSISTEMAS DE AXIS

La arquitectura de Axis consiste de varios subsistemas, por simplicidad estos subsistemas están encargados de procesar mensajes la forma de hacerlo es por medio de un conjunto de manipuladores (Handlers)



Composición de Apache Axis

El diagrama anterior muestra una idea muy general la capa de los subsistemas de Axis, las capas de abajo son independientes de las de arriba

En la capa de admin. El archivo WSDD (Deployment class Simple Provider) que es un archivo XML para descriptores de despliegue que configuran el motor de axis
SOAP consiste de una cubierta de un encabezado y un cuerpo m dentro de ellos se definen elementos que contienen nombres de espacios (namespaces) que refieren a URL o o un nombre local, y codifican estilos, un estándar único del cual es definido por SOAP

La capa de decodificación hace el proceso de trasformación entre los valores de los tipos de datos de un lenguaje de programación y su representación en XML. En Axis, significa codificar (serializing) objetos Java y primitivas en XML y decodificar (o deserializaing) en objetos Java y primitivas. Las clases básicas que implementan esos pasos son serializers y deserializers.

ANEXO VII. INSTALACIÓN DE LA APLICACIÓN DUKE´S BANK EN EL SERVIDOR DE APLICACIÓN JBOSS

Antecedente
Bajar el servidor en la dirección correspondiente:

Http://www.jboss.org/
http://www.sourceforge.com/

Los archivos binarios se encuentran en extensiones .zip, tar.gz, bz2
Como primer paso se tiene que instalar el servidor de JBoss por lo cual se establece la carpeta en donde se encuentra ubicado

Opt/jboss/jboss3.2.1/

Declarar en al archivo /etc/profile la ubicación del servidor

JBOSS_HOME= /opt/jboss/jboss3.2.1

De igual forma la versión de java

JAVA_HOME=/jdk/java

Y su correspondiente classpath

CLASSPATH:./JAVA_HOME/lib/tools.jar

Para comprobar la instalación de JBoss hay que ubicarse en la carpeta JBOSS_HOME/bin
Donde se encuentra el archivo run.sh para versiones de Unix y run.bat para Windows.
Aquí cabe mencionar que eñl servidro `puede correr de acuerdo al número de servicios que se desean levantar: minimal, defaul y all

Para caso de el proyecto se inicializará el servidro con todos los servicios yar que este cuenta con el servicio para WebService entre otros.

En particular se puede personalizar la configuración del servidor , por lo que se copiar los archivos de la carpeta JBOSS_HOME/server/all en JBOSS_HOME/server/bank esta última carpeta es creada por el usuario quien puede personalizar la configuración de su servidor para una aplicación en especifico.

Por lo que podemos iniciar el servidor como:

#./run.sh –c deploy

donde la bandera –c indica el tipo de configuración en el cual trabajará JBoss.

Para comprobar que el servidor ha sido levantado dirigirse a la dirección http://localhost:8080/jmx-console/

En esta página se despliegan los Mbean los cuales se compone el servidor.

La aplicación requiere de Apache Ant por lo cual se ha de trabajar con la versión ant 6.0. :

Ubicar en el profile el path donde se encuentra la versión de ant.
En esta caso ubicado en /user/local/bin/ant

Para utilizar Ant este requiere de un script el cual es un descriptor llamado build.xml, en el proyecto se ha creado otro con el nombre de jboss-build.xml

Archivo jboss-build.xml

```
project name="jboss-dukes-bank" default="about" basedir=".">

 <property file="${basedir}/../build.properties"/>
 <property name="src.dir"       value="${basedir}/src"/>
 <property name="build.dir"     value="${basedir}/build"/>
 <property name="servlet.jar"   value="${jboss.server}/lib/javax.servlet.jar"/>

 <!--
  | Declaracion de classpath para correr el  client
  -->
 <path id="client.classpath">
   <pathelement location="${build.dir}"/>
   <fileset dir="${jboss.home}/client">
     <include name="**/*.jar"/>
   </fileset>
   <!-- the following jars are only used for the web-service client -->
   <fileset dir="${jboss.home}/server/all/deploy/jboss-net.sar">
     <include name="*.jar"/>
   </fileset>
 </path>

 <!--
  | La construccion del classpath
  -->
 <path id="build.classpath">
           <path id="axis.classpath">
      <pathelement location="${build.dir}"/>
           <fileset dir="/axis-1_1/lib">
          <include name="*.jar"/>
    </fileset>
           <fileset dir="${jboss.server}/lib/">
          <include name="javax.servlet*.jar"/>
     </fileset>
   </path>



   <path refid="client.classpath"/>
   <pathelement location="${servlet.jar}"/>
 </path>

 <!--
  | Hypersonic SQL classpath
  -->
 <path id="hsql.classpath">
   <pathelement location="${jboss.server}/lib/hsqldb.jar"/>
```

```
</path>

<!-- ===================================================================== -->
<!—Inicializa Build                                    -->
<!-- ===================================================================== -->
<target name="prepare">
  <mkdir dir="${build.dir}"/>
</target>

<!-- ===================================================================== -->
<!—Compila el código fuente                            -->
<!-- ===================================================================== -->
<target name="compile" depends="prepare" >
  <javac destdir="${build.dir}" classpathref="build.classpath">
    <src path="${src.dir}"/>
  </javac>
</target>

<target name="package-account" depends="compile">
  <delete file="jar/account-ejb.jar"/>
  <copy file="dd/account-ejb.xml" tofile="build/ejb-jar.xml" overwrite="true"/>
  <copy file="dd/account-jboss.xml" tofile="build/jboss.xml" overwrite="true"/>

  <jar jarfile="jar/account-ejb.jar">
    <metainf dir="${build.dir}" includes="ejb-jar.xml,jboss.xml" />
    <fileset dir="${build.dir}">
      <include name="com/sun/ebank/ejb/account/**" />
      <include name="com/sun/ebank/ejb/exception/**" />
      <include name="com/sun/ebank/util/**" />
      <include name="com/sun/ebank/ejb/customer/Customer.class" />
      <include name="com/sun/ebank/ejb/customer/CustomerHome.class" />
    </fileset>
  </jar>
</target>

<target name="package-customer" depends="compile">
  <delete file="jar/customer-ejb.jar"/>
  <copy file="dd/customer-ejb.xml" tofile="build/ejb-jar.xml" overwrite="true"/>
  <copy file="dd/customer-jboss.xml" tofile="build/jboss.xml" overwrite="true"/>

  <jar jarfile="jar/customer-ejb.jar">
    <metainf dir="${build.dir}" includes="ejb-jar.xml,jboss.xml" />
    <fileset dir="${build.dir}">
      <include name="com/sun/ebank/ejb/customer/**" />
      <include name="com/sun/ebank/ejb/exception/**" />
      <include name="com/sun/ebank/util/**" />
    </fileset>
  </jar>
</target>

<target name="package-tx" depends="compile">
  <delete file="jar/tx-ejb.jar"/>
  <copy file="dd/tx-ejb.xml" tofile="build/ejb-jar.xml" overwrite="true"/>
  <copy file="dd/tx-jboss.xml" tofile="build/jboss.xml" overwrite="true"/>

  <jar jarfile="jar/tx-ejb.jar">
```

```
    <metainf dir="${build.dir}" includes="ejb-jar.xml,jboss.xml" />
    <fileset dir="${build.dir}">
      <include name="com/sun/ebank/ejb/tx/**" />
      <include name="com/sun/ebank/ejb/exception/**" />
      <include name="com/sun/ebank/util/**" />
      <include name="com/sun/ebank/ejb/customer/Account.class" />
      <include name="com/sun/ebank/ejb/customer/AccountHome.class" />
    </fileset>
  </jar>
</target>

<target name="package-ejb" depends="package-account,package-customer,package-tx" />

<target name="package-client" depends="compile">
  <copy todir="${build.dir}" >
    <fileset dir="${src.dir}">
      <include name="**/appclient/*.properties" />
    </fileset>
    <mapper type="flatten" />
  </copy>
  <delete file="jar/app-client.jar"/>
  <copy file="dd/jboss-client.xml" todir="${build.dir}" overwrite="true"/>
  <copy file="dd/app-client.xml" tofile="${build.dir}/application-client.xml" overwrite="true"/>
  <copy file="dd/jndi.properties" todir="${build.dir}"/>

  <jar jarfile="jar/app-client.jar">
    <metainf dir="${build.dir}" includes="jboss-client.xml,application-client.xml" />
    <fileset dir="${build.dir}">
      <include name="jndi.properties"/>
      <include name="com/sun/ebank/appclient/**" />
      <include name="Admin*.properties" />
      <include name="com/sun/ebank/ejb/exception/**" />
      <include name="com/sun/ebank/util/**" />
      <include name="com/sun/ebank/ejb/customer/Account.class" />
      <include name="com/sun/ebank/ejb/customer/AccountHome.class" />
    </fileset>
  </jar>
</target>


<target name="package-web" depends="compile">
   <delete file="jar/web-client.war"/>
   <war warfile="jar/web-client.war" webxml="dd/web.xml">
     <fileset dir="${src.dir}/web" >
       <include name="*.jsp" />
       <include name="images/*.gif" />
     </fileset>
     <webinf dir="dd">
        <include name="jboss-web.xml" />
     </webinf>
     <webinf dir="${src.dir}/web" >
        <include name="*.tld" />
        <exclude name="*.jsp" />
        <exclude name="*.txt" />
        <exclude name="images/**" />
      </webinf>
```

```
    <webinf dir="jar">
      <include name="*.tld" />
    </webinf>
    <lib dir="jar">
      <include name="struts.jar" />
      <include name="commons*.jar"/>
     </lib>
    <classes dir="${build.dir}" >
      <include name="**/*.class" />
      <exclude name="com/sun/ebank/appclient/**" />
      <exclude name="com/sun/ebank/ejb/**" />
      <exclude name="com/sun/ebank/util/**" />
    </classes>
    <classes dir="${src.dir}/web" >
      <include name="*.properties" />
    </classes>
  </war>
</target>

<!--
 | Crea  el archvo  file contiene ejb jars y el cliente web  war.
 -->
<target name="assemble-app">
  <delete file="jar/JBossDukesBank.ear"/>
  <ear destfile="jar/JBossDukesBank.ear" appxml="dd/application.xml">
    <fileset dir="jar" includes="*-ejb.jar,app-client.jar,*.war,*.wsr"/>
    <fileset dir="src" includes="users.properties,roles.properties"/>
  </ear>
</target>

<!--
 | Despliega el archvio EAR lo remueve y lo copia al directorio  deploy de JBoss.
 -->
<target name="deploy" depends="assemble-app">
  <copy file="jar/JBossDukesBank.ear" todir="${jboss.server}/deploy"/>
</target>


<!--
 | Corre la aplicación cliente.
 -->
<target name = "run-client">
  <echo>${java.class.path}</echo>
  <java classname="com.sun.ebank.appclient.BankAdmin" fork="yes">
    <classpath>
      <pathelement path="jar/app-client.jar"/>
      <path refid="client.classpath"/>
      <pathelement path="${java.class.path}"/>
    </classpath>
  </java>
</target>


<!--
 | Llama a la utileria HSQL.
 -->
```

```
<target name="db-create-table">
  <java classname="org.hsqldb.util.ScriptTool" fork="yes" >
    <arg value="-url"/>
    <arg value="jdbc:hsqldb:hsql:"/>
    <arg value="-database"/>
    <arg value="//localhost:1701"/>
    <arg value="-script"/>
    <arg value="sql/hsql-create-table.sql"/>
    <classpath refid="hsql.classpath"/>
  </java>
</target>

<target name="db-insert">
  <java classname="org.hsqldb.util.ScriptTool" fork="yes" >
    <arg value="-url"/>
    <arg value="jdbc:hsqldb:hsql:"/>
    <arg value="-database"/>
    <arg value="//localhost:1701"/>
    <arg value="-script"/>
    <arg value="sql/insert.sql"/>
    <classpath refid="hsql.classpath" />
  </java>
</target>

<target name="db-list">
  <java classname="org.hsqldb.util.ScriptTool" fork="yes" >
    <arg value="-url"/>
    <arg value="jdbc:hsqldb:hsql:"/>
    <arg value="-database"/>
    <arg value="//localhost:1701"/>
    <arg value="-script"/>
    <arg value="sql/listAccount.sql" />
    <classpath refid="hsql.classpath" />
  </java>
</target>

<target name="db-delete">
  <java classname="org.hsqldb.util.ScriptTool" fork="yes" >
    <arg value="-url"/>
    <arg value="jdbc:hsqldb:hsql:"/>
    <arg value="-database"/>
    <arg value="//localhost:1701"/>
    <arg value="-script"/>
    <arg value="sql/delete.sql" />
    <classpath refid="hsql.classpath" />
  </java>
</target>

<target name="db-reset-key">
  <java classname="org.hsqldb.util.ScriptTool" fork="yes" >
    <arg value="-url"/>
    <arg value="jdbc:hsqldb:hsql:"/>
    <arg value="-database"/>
    <arg value="//localhost:1701"/>
    <arg value="-script"/>
    <arg value="sql/hsql-reset-key.sql" />
```

```
      <classpath refid="hsql.classpath" />
    </java>
  </target>

<!--
 | Contruir el   web-service WSR file para JBoss-Net
 -->
  <target name = "wsr">
    <delete file="jar/ebank.wsr"/>
    <jar jarfile="jar/ebank.wsr" basedir="${build.dir}">
      <metainf dir="dd" includes="web-service.xml" />
      <exclude name="**"/>
    </jar>
  </target>

  <target name="deploy-wsr" depends="wsr">
    <copy file="jar/ebank.wsr" todir="${jboss.server}/deploy"/>
  </target>

<!--
 | Correr el cliente WebServices
 -->
  <target name = "run-ws-client">
    <echo>${java.class.path}</echo>
    <java classname="WSClient">
      <classpath>
        <path refid="client.classpath"/>
        <pathelement path="${java.class.path}"/>
      </classpath>
    </java>
  </target>
<-- Crear el Webservice en JBossWS  con la herramienta Java2WSDL de Axis
-->
  <target name="wsdl">
      <mkdir dir="dd/ws/wsdl" />

      <java classname="org.apache.axis.wsdl.Java2WSDL"
              classpathref="axis.classpath" fork="yes">
          <arg value="-lhttp://doriana:8080/bankws-ejb/Teller" />
          <arg value="-STellerService" />
          <arg value="-sTellerEndpoint" />
          <arg value="-odd/ws/wsdl/teller.wsdl" />
          <arg value="-uENCODED" />
          <arg value="com.jboss.ebank.TellerEndpoint" />
      </java>
    </target>

<target name="deploy-ws">
      <copy file="jar/bankws-ejb.jar" todir="${jboss.server}/deploy"/>
    </target>

    <target name="run-ws">
      <java classname="com.jboss.ebank.WSClient" fork="true">
        <classpath>
          <path refid="client.classpath"/>
          <pathelement path="${java.class.path}"/>
```

```
        </classpath>
      </java>
   </target>
         <target name="package-ws">
      <mkdir dir="jar" />
      <delete file="jar/bankws-ejb.jar"/>

      <jar jarfile="jar/bankws-ejb.jar">
         <metainf dir="dd/ws"  includes="**/*" />

         <fileset dir="${build.dir}">
            <include name="com/jboss/ebank/**" />
            <include name="com/sun/ebank/ejb/account/AccountController**"/>
            <include name="com/sun/ebank/ejb/exception/**"/>
            <include name="com/sun/ebank/util/**"/>
         </fileset>
      </jar>
   </target>



<!--
 | Correr la utilería  Axis TCPMon
 -->
  <target name = "tcpmon">
    <echo>${java.class.path}</echo>
    <java classname="org.apache.axis.utils.tcpmon" fork="true">
      <classpath>
        <path refid="client.classpath"/>
        <pathelement path="${java.class.path}"/>
      </classpath>
    </java>
  </target>

  <target name="clean">
    <delete dir="${build.dir}" />
  </target>

</project>
```

UBICACIÓN DEL PROYECTO:

/home/bank/j2eetutorial/bank

En este misma ubicación se encuentra el el script jboss-buil.xml. Pero antes de ello se ha de modificar el archivo build.properties y editar en la propiedad jboss.home el path donde se encuentra ubicada la configuración del servidor:

jboss.home=JBOSS_HOME/server/bank

de esta forma la aplicación esta preparada para comenzar el proceso de compilación de los archivos fuentes así como también crear, borrar, copiar y desplegar el archivo .ear (enterprise archive) que será el archivo que contiene el empaquetamiento global de la aplicación

Compilación

En la linea de comando

#ant –f jboss-buil.xml compile

Empaquetamiento de los componentes web el cual creará el archivo .war

#ant –f jboss-build.xml paxkage-web

Empaquetamiento de los componentes ejb el cualñ crea un archivo .jar

#ant –f jbosss-build.xml package-ejb

Empaquetamiento de cliente java

#ant –f jboss-build.xml package-cliente

Ensamblaje de la aplicación

#ant –f jboss-build.xml assembled-app

Despliegue de la aplicación

#ant –f jboss-build.xml deploy

Nota: se recomienda que para el despliegue de la aplicación ya se encuentre el servidor corriendo si ningún tipo de error.

Se aprecia que la aplicación esta correctamente funcionando visitando: http://localhost:8080/bank/main

ANEXO VIII. IMPLEMENTACION DEL SESSION BEAN TELLER COMO WEBSERVICE EN JBOSS

Para el despliegue del WebService aprovechando el archivo

Crear el archivo WSR (Web Service Archve)

#ant –f jboss-build.xml wsr

Desplegar el archivo WSR

#ant –f jboss-build.xml deploy-wsr

Crear un programa cliente

#ant –f jboss-build.xml run-ws

El programa cliente para desplegar es el siguiente

Programa Cliente en lenguaje Java

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import com.sun.ebank.util.AccountDetails;
import javax.xml.namespace.QName;



public class WSClient
{
  public static void main(String [] args) throws Exception
  {
    // The connection URL. Change the port number if you are using TCPMON
    String endpoint = "http://localhost:8080/jboss-net/services/AccountController";

    Service service = new Service();
    Call    call    = (Call) service.createCall();

    call.setTargetEndpointAddress( new java.net.URL(endpoint) );
    call.setOperationName("getDetails");

//    call.addParameter("accountId", org.apache.axis.Constants.XSD_STRING, ParameterMode.IN);
    call.setReturnClass(AccountDetails.class);

    QName qn = new QName("http://net.jboss.org/bank","AccountDetails");
```

```
    call.registerTypeMapping(AccountDetails.class, qn,
            new org.apache.axis.encoding.ser.BeanSerializerFactory(AccountDetails.class, qn),
            new org.apache.axis.encoding.ser.BeanDeserializerFactory(AccountDetails.class, qn));

    AccountDetails ret = (AccountDetails) call.invoke( new Object[] { "5005" } );

    System.out.println(ret.getDescription() + " " + ret.getType());
  }
}
```

ANEXO IX. IMPLEMENTACIÓN DEL WEB SERVICE TELLER CORRIENDO EN EL SERVIDOR JBOSS-4.0.1

Haciendo uso Ant que hace uso una vez más del archivo jboss-build.xml se determina los siguientes comandos:

El primer paso es compilar el código en este caso TellerBean esta expuesto en la carpeta J2EETUTORIAL/src/com/jboss/ebank/, así como otras clases como Array_Of_String. Java, TellerEndpoint.java, y el programa cliente del webService WSCliente.java

TellerBean

```
package com.jboss.ebank;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Iterator;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.rmi.RemoteException;

import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.ejb.CreateException;

import com.sun.ebank.ejb.account.AccountController;
import com.sun.ebank.ejb.account.AccountControllerHome;
import com.sun.ebank.util.AccountDetails;

public class TellerBean
   implements SessionBean
{
   public TellerBean() {
   }

   private AccountController getController()
         throws RemoteException,
              NamingException,
              CreateException
   {
         InitialContext ctx = new InitialContext();
         AccountControllerHome home = (AccountControllerHome)
            ctx.lookup("java:comp/env/ejb/account");

         return home.create();
   }

   public BigDecimal getAccountBalance(String accountID) {
         try {
            AccountController mgr     = getController();
            AccountDetails    details = mgr.getDetails(accountID);

            return details.getBalance();
         } catch (Exception e) {
```

```
            e.printStackTrace();
            return new BigDecimal(0);
        }
    }

    public String[] getAccountsOfCustomer(String customerId)
    {
        try {
            AccountController mgr = getController();

            ArrayList list    = getController().getAccountsOfCustomer(customerId);

            String[]  res     = new String[list.size()];
            Iterator  it_list = list.iterator();
            for (int i=0; it_list.hasNext(); i++) {
                AccountDetails details = (AccountDetails) it_list.next();
                res[i] = details.getAccountId();
            }

            return res;
        } catch (Exception e) {
            e.printStackTrace();
            return new String[0];
        }
    }


    public void ejbCreate() { }
    public void ejbRemove() { }

    public void ejbActivate()  { }
    public void ejbPassivate() { }

    public void setSessionContext(SessionContext sc) { }
}
```

TellerEnpoint.java

```
package com.jboss.ebank;

import java.rmi.Remote;
import java.rmi.RemoteException;

import java.util.ArrayList;

import java.math.BigDecimal;

public interface TellerEndpoint
    extends Remote
{
    public String[] getAccountsOfCustomer(String customerId)
        throws RemoteException;

    public BigDecimal getAccountBalance(String accountID)
        throws RemoteException;
```

```
}
```

ArrayOf_xsd_string.java

```
package com.jboss.ebank;

public class ArrayOf_xsd_string
{




}
```

#ant –f jboss-buil.xml compile

El comando siguiente crea la página wsdl del servicio así como el lugar donde se ubica. A su vez este comando hace uso de la herramienta de Axis java2wsdl la cual viene integrada en JBossWS

#ant –f jboss-buil.xml wsdl

Teller.wsdl

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://ebank.jboss.com"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://ebank.jboss.com" xmlns:intf="http://ebank.jboss.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <wsdl:types>
- <schema targetNamespace="http://ebank.jboss.com"
xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
- <complexType name="ArrayOf_xsd_string">
- <complexContent>
- <restriction base="soapenc:Array">
<attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
</restriction>
</complexContent>
</complexType>
</schema>
</wsdl:types>
- <wsdl:message name="getAccountsOfCustomerResponse">
<wsdl:part name="getAccountsOfCustomerReturn" type="impl:ArrayOf_xsd_string" />
</wsdl:message>
- <wsdl:message name="getAccountsOfCustomerRequest">
<wsdl:part name="in0" type="xsd:string" />
</wsdl:message>
- <wsdl:message name="getAccountBalanceRequest">
<wsdl:part name="in0" type="xsd:string" />
```

```
  </wsdl:message>
- <wsdl:message name="getAccountBalanceResponse">
  <wsdl:part name="getAccountBalanceReturn" type="xsd:decimal" />
  </wsdl:message>
- <wsdl:portType name="TellerEndpoint">
- <wsdl:operation name="getAccountsOfCustomer" parameterOrder="in0">
  <wsdl:input message="impl:getAccountsOfCustomerRequest"
name="getAccountsOfCustomerRequest" />
  <wsdl:output message="impl:getAccountsOfCustomerResponse"
name="getAccountsOfCustomerResponse" />
  </wsdl:operation>
- <wsdl:operation name="getAccountBalance" parameterOrder="in0">
  <wsdl:input message="impl:getAccountBalanceRequest" name="getAccountBalanceRequest" />
  <wsdl:output message="impl:getAccountBalanceResponse" name="getAccountBalanceResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="TellerEndpointSoapBinding" type="impl:TellerEndpoint">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="getAccountsOfCustomer">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="getAccountsOfCustomerRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://ebank.jboss.com" use="encoded" />
  </wsdl:input>
- <wsdl:output name="getAccountsOfCustomerResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://ebank.jboss.com" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getAccountBalance">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="getAccountBalanceRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://ebank.jboss.com" use="encoded" />
  </wsdl:input>
- <wsdl:output name="getAccountBalanceResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://ebank.jboss.com" use="encoded" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="TellerService">
- <wsdl:port binding="impl:TellerEndpointSoapBinding" name="TellerEndpoint">
  <wsdlsoap:address location="http://DORIANA:8080/bankws-ejb/Teller" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

Cliente VisualBasic.Net

Cliente Visual Basic Studio

```
Public Class Form1
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents txtcuenta As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents lblresult As System.Windows.Forms.Label
    Friend WithEvents btnconectarse As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Me.txtcuenta = New System.Windows.Forms.TextBox
        Me.Label1 = New System.Windows.Forms.Label
        Me.lblresult = New System.Windows.Forms.Label
        Me.btnconectarse = New System.Windows.Forms.Button
        Me.SuspendLayout()
        '
        'txtcuenta
        '
        Me.txtcuenta.Location = New System.Drawing.Point(208, 120)
        Me.txtcuenta.Name = "txtcuenta"
        Me.txtcuenta.TabIndex = 0
        Me.txtcuenta.Text = ""
        Me.txtcuenta.TextAlign = System.Windows.Forms.HorizontalAlignment.Center
        '
        'Label1
        '
```

```
        Me.Label1.BackColor = System.Drawing.Color.LightBlue
        Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 14.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.Location = New System.Drawing.Point(152, 64)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(240, 23)
        Me.Label1.TabIndex = 1
        Me.Label1.Text = "Cliente WebService Teller"
        Me.Label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
        '
        'lblresult
        '
        Me.lblresult.BackColor = System.Drawing.Color.Beige
        Me.lblresult.Location = New System.Drawing.Point(144, 232)
        Me.lblresult.Name = "lblresult"
        Me.lblresult.Size = New System.Drawing.Size(232, 80)
        Me.lblresult.TabIndex = 2
        '
        'btnconectarse
        '
        Me.btnconectarse.BackColor = System.Drawing.Color.Beige
        Me.btnconectarse.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.btnconectarse.ForeColor = System.Drawing.SystemColors.ActiveCaption
        Me.btnconectarse.Location = New System.Drawing.Point(224, 168)
        Me.btnconectarse.Name = "btnconectarse"
        Me.btnconectarse.TabIndex = 3
        Me.btnconectarse.Text = "conexion"
        Me.btnconectarse.TextAlign = System.Drawing.ContentAlignment.TopCenter
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.BackColor = System.Drawing.Color.Khaki
        Me.ClientSize = New System.Drawing.Size(504, 350)
        Me.Controls.Add(Me.btnconectarse)
        Me.Controls.Add(Me.lblresult)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.txtcuenta)
        Me.Name = "Form1"
        Me.Text = "Form1"
        Me.ResumeLayout(False)

    End Sub

#End Region

    Private Sub btnconectarse_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnconectarse.Click

        Dim objWebRef As New localhost.TellerService
        Dim i As Integer


        Dim strResult() = objWebRef.getAccountsOfCustomer(txtcuenta.Text)
```

```
    For i = 0 To strResult.Length
        lblresult.Text = objWebRef.getAccountBalance(strResult(i))
    Next i

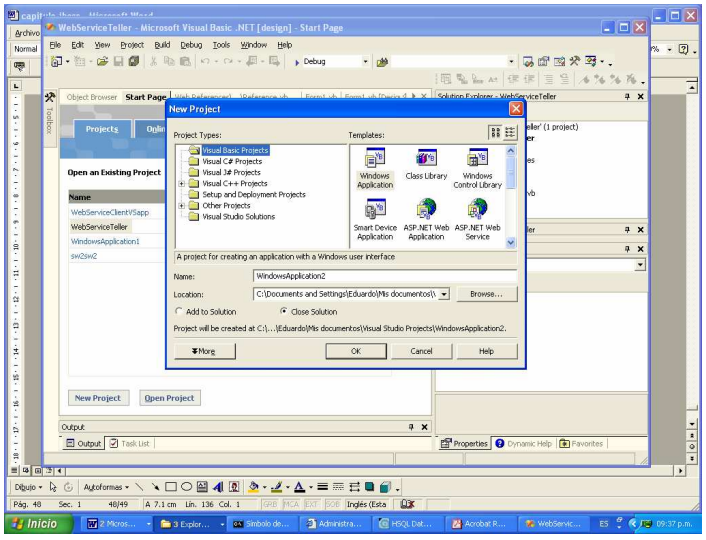


  End Sub

  Private Sub lblresult_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
lblresult.Click

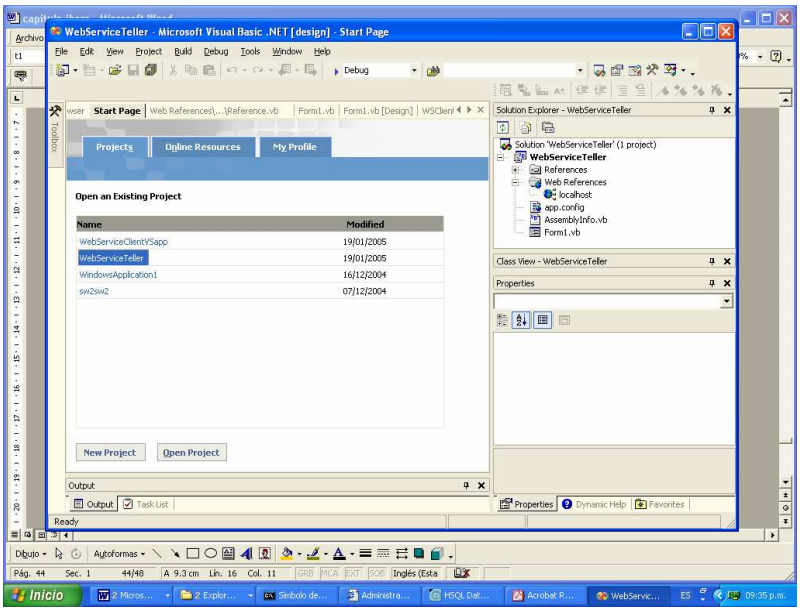
  End Sub
End Class
```

Para crear la aplicación cliente que se comunique con Teller Webservice requerimos en Visual Basic Studio Abrir un nuevo proyecto el cual se le ha de llamar WebServiceTeller. Entonces colocar el nombre de esta misma en Name



Abrir Nuevo Proyecto Visual Studio.Net

En la figura siguiente se encuentra un explorador del proyecto.
Indicar en Webreferences el recurso disponible en la web con que se trabajará la aplicación cliente

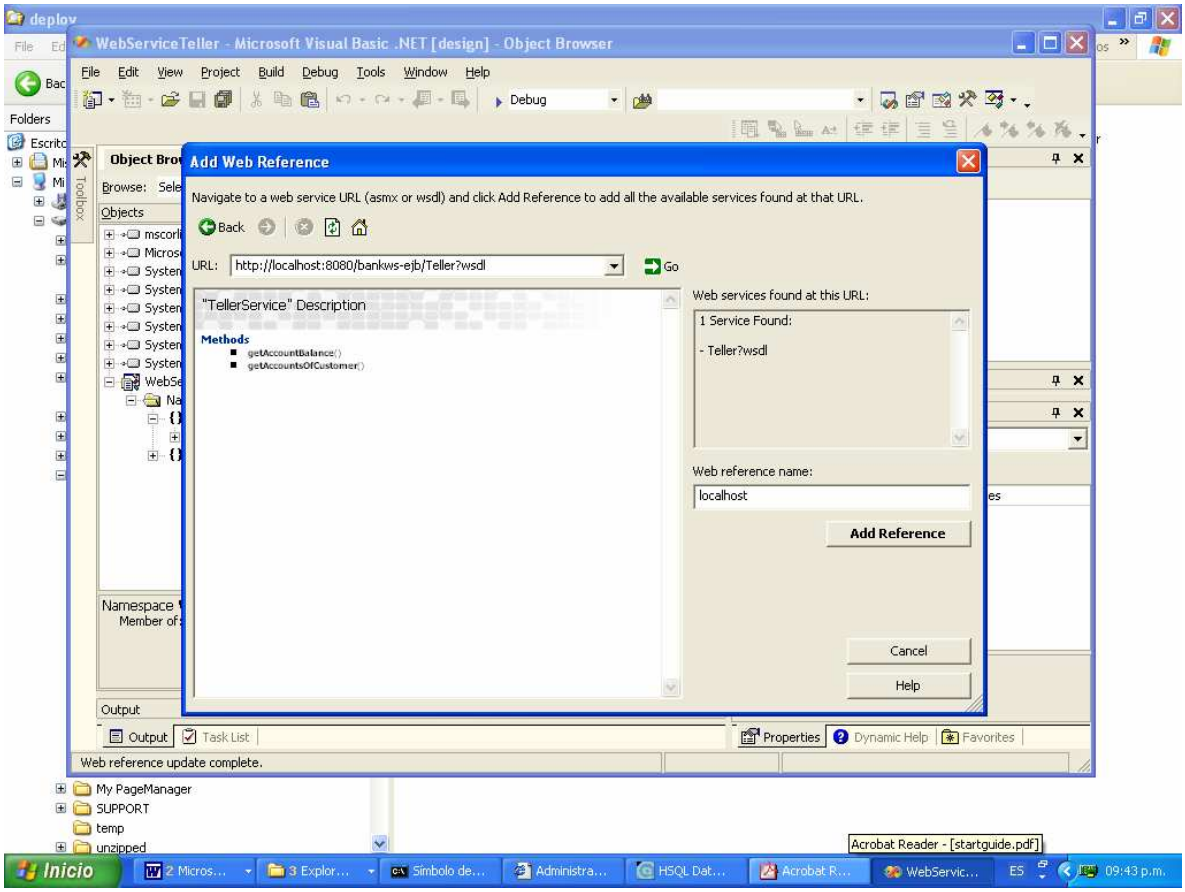


Entorno Visual Basic Studio .Net WebServices

Declarar en AddWebReferences la dirección donde se encuentra la ubicación del WebService, es decir, de la página Teller.wsdl

http://doriana:8080/bankws-ejb/Teller?wsdl

En la casilla AddReferences escribir cualquier nombre que identifique este modulo o referencia, en este caso el nombre de esta es localhost.



Localizaciòn WebServices Disponibles

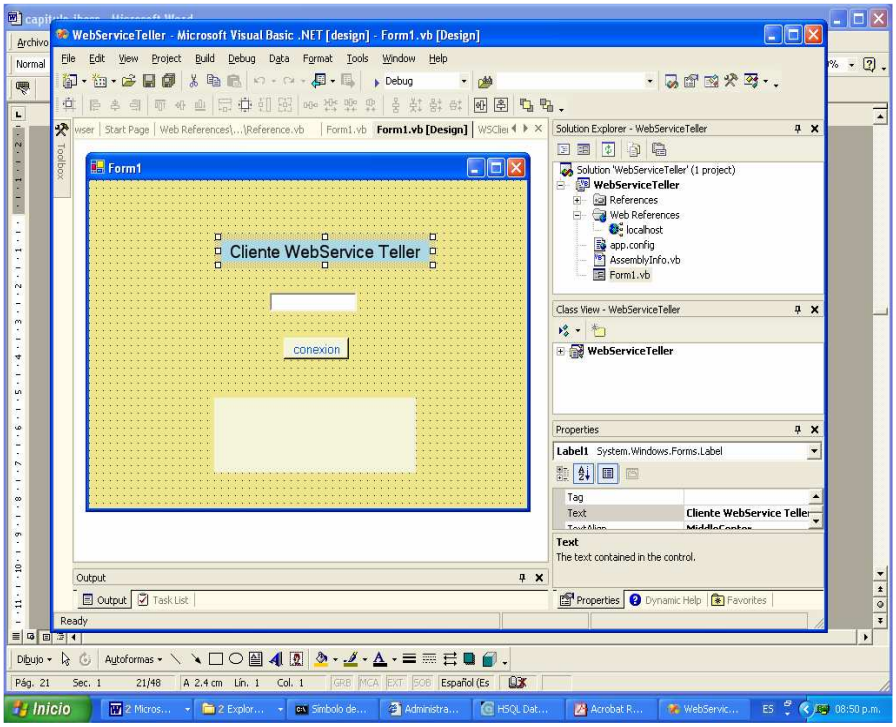
Posteriormente crear la forma o la interfase gráfica del cliente web service para ello es necesario incluir de la barra de herramienta o ToolBox los controles que muestra la figura siguiente:

Label1: Label1

Textbox: txtcuenta

Botton: btnconectarse

Label2: lblresult



Creación de la forma WebServiceTeller

GLOSARIO DE TÉRMINOS

CORBA (Common Object request Architecture).

XML (Extensible Markup Language) Base con que se conforma un Web Service, y a partir de esta surge un estándar capaz de manipular información y comunicarse de una forma más efectiva entre los sistemas de información que hacen uso de datos para diferentes propósitos.

TI Tecnologías de Información (TI) basadas en Internet.

CE Conceptos electrónicos (CE) que se manejan en el ambiente electrónico, los cuales se ven inmersos en los negocios electrónicos (e-business, e-commerce, business-to-business, etc.), empresas no lucrativas y organizaciones no gubernamentales.

SBI Servicios basados en Internet (SBI) que utilizan tanto dentro y fuera de su negocio, para lograr una diferenciación en el mercado.

RPC (Remote Procedure Calls Llamados de procedimiento remotos) a una función remota ser llamada como si se tratara de una local.

WEBSERVICES Tecnología orientada a objetos, y representan una revolución al alejarse de las arquitecturas tradicionales tipo *cliente-servidor* a nuevas arquitecturas distribuidas tipo *igual-a-igual* (*peer-to-peer).*

Proveedor de Servicio Desde el punto de vista de arquitectura, es la plataforma que provee el servicio.

Registro de Servicios Es un depósito de descripciones de servicios que puede ser consultado, donde los proveedores de servicios publican sus servicios y los solicitantes encuentran los servicios y detalles para utilizar dichos servicios.

Solicitante de servicios Desde el punto de vista de la arquitectura, la aplicación o cliente que busca e invoca un servicio.

Bind Los solicitantes de servicios negocian con los proveedores de servicios para acceder e invocar servicios comerciales (*e*-business).

WSDL (Web Services Definition Service) Este protocolo se encarga de describir el web service cuando es publicado. Es el lenguaje XML que los proveedores emplean para describir sus web services.

SOAP (Simple Object Access Protocol) Permite que programas que corren en diferentes sistemas operativos se comuniquen. La comunicación entre las diferentes entidades se realiza mediante mensajes que son ruteados mediante el protocolo SOAP.

UDDI (Universal Description Discovery and Integration) Este protocolo permite la publicación y localización de los servicios. Los directorios UDDI actúan como una guía telefónica de web services.

URN Nombre de recurso unificado (unified resource name) excepcionalmente identifica el servicio a clientes.

DOCUMENT Conocido también como estilo de mensaje-orientado. Este proporciona una capa baja de abstracción y requiere mas trabajo de programación.

WSIL Un documento de lenguaje de inspección Web Services (WSIL) provee información de localización para invocar Web Services.

Types (Tipos) Un contenedor para la definición de tipos de datos usando algunos tipos de sistemas tales como el esquema XML.

Message (Mensaje) Un abstracto, tipo de definición de los datos que son comunicados. Un mensaje puede tener uno o más tipos de componentes.

Port type (Tipo de Puerto) Un conjunto abstracto de una o mas operaciones soportados por uno o mas puertos. Cada una de las operaciones define una entrada y una salida de mensaje así como es un mensaje opcional por default.

Operation (Operacion).- Una descripción abstracta de una acción soportada por el servicio.

Binding Un protocolo concreto y especificación de formato de datos para un particular tipo de puertos. La información binding contiene el nombre del protocolo, el estilo de invocación, un servicio ID, y la codificación para cada operación.

Port (Puerto) Un simple punto final, que es definido como una agregación de un binding y una dirección de red.

Service (Servicio).- Una colección de puertos relacionados.

Business entity (Entidad de Negocio) La lista de entidades de negocio es similar a las paginas blancas y amarillas. Una entidad de negocio describe una compañía u organización. Estas entidades suministran información tal como el nombre del negocio, contactos, descripciones, identificadores y categorización.

Business service (Servicio de Negocios) Un servicio de negocio es un contenedor descriptivo usado por un grupo fijo de Web Services relacionados para un proceso de negocio o grupo de servicios.

Binding template Este servicio contiene información de acceso. Allí se encuentra la descripción técnica de un Web Service relevante para los desarrolladores de aplicaciones quienes desean encontrar e invocar un Web Service.

tModel Un tModel (modelo técnico) es técnicamente una impresión digital que contiene meta-datos acerca del tipo de especificaciones así como es la categorización de la información.

Taxonomy (Taxonomia) Una taxonomía es un esquema para categorización.

Publisher assertions (relación entre negocios) También llamado esto relación entre negocios. Aquí son diferentes tipos de relaciones: padre-hijo, par-par, e identidad. Esto hace posible un modelo de negocios complejo.

EJB Un enterprise java bean es un componente del servidor que encapsula la lógica del negocio en la aplicación, son componentes portables.

Session Bean Representa un solo cliente dentro del servidor J2EE.

Entity Bean Representa el objeto en el negocio en un mecanismo de persistencia y esta es una de las características mas, permite un acceso compartido, tener llaves primarias, y participa en la relación con otros entity beans.

Message-Driven Bean Procesa mensajes asincrónicamente los mensajes pueden ser enviados a cualquier componente J2EE y aplicación cliente, a otro enterprise bean o a un componente Web o por medio de una aplicación JMS (Java Message Service) o sistema que no use la tecnología J2EE.

WSDD Web Service Deployment Desciptor.

JNDI Java Name Directory Interfase.

BIBLIOGRAFÍA

[1] I. Foster and C. Kesselman. ***The Grid: Blueprint for a New Computing Infrastructure*****. Morgan Kaufmann, 1999.**
[2] I. Foster, C. Kesselman, J.Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. service infrastructure wg, global grid forum. http://www.globus.org/research/papers/ogsa.pdf.
[3] P. Sherad.
Grid architecture blueprint builds on web services, February 2002.
http://www.internetnews.com/xSP/article.php/973221.
[4] O. Storz, P.V. Boddupalli, N. Davies, A. Friday, and M. Wu. Leveraging the grid to provide a global platform for ubiquitous computing research. Technical Report CSEG/2/03, University of Lancaster, 2003.
[5] R. Zade and A. Moharil. Building a business logic layer over multiple web services: Leveraging multiple web services to build a truly distributed web services architecture. ***Web Services Journal*****, 3(8), 2003.**

[6] Clempner, Kerik Julio y Gutiérrez, Tornés Agustín, Administración y Ejecución de un Plan Estratégico de Tecnología de Información [texto electrónico], Revista Digital Universitaria, URL: http://www.revista.unam.mx/vol.3/num1/art1/ (Accesado Mayo 12, 2002), Publicación en Dirección General de Servicios de Cómputo Académico-UNAM
Ciudad Universitaria, México D.F. 31 de marzo del 2002 Vol. 3 No. 1.

[7] TURBAN Efraim, Lee Jae, King David, Chung H. Michael, Electronic Commerce, a managerial perspective, Prentice Hall, 2000

[8] PORTER Michael, On Competition, Harvard Business School Publishing, Boston, MA, 1998

[9] LAUDON Kennet C., Laudon Jane Price, Essentials of Management Information Systems, Transforming Business and Management, Prentice Hall, New Jersey, July 2001.

[10] O'BRIEN James A., Management Information Systems. Managing Information Technology in the E-Business Enterprise, Fifth Edition, McGraw-Hill, 2002

PÁGINAS VISITADAS Y ARTICULOS

http;//java.sun.com/
http://www.ibm.com/
http://www.jboss.org/

Artículo Net versus J2EE Web Services
http://webservicesarchitec.com/content/articles/

Articulo Web Services la siguiente generación de Internet
http://revista.robotiker.com/articulos/articulo62/pagina1.jsp

Articulo: Brave New Apps: The Application Servers
By Richard V. Dragan
Revista PCMagazine September 2, 2003
http://www.pcmag.com/article2/0,4149,1218682,00.asp

http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/peservices.htm

http://www.ebstudio.com/download/WebStudio/Web_Services.pdf

http://www.improven-consultores.com/paginas/documentos_gratuitos/XML_negocios.phpArtiuclo de informatizate

http://www.pcm.gob.pe/portal_ongei/publicaciones/cultura/Lib5038/glosa.HTM

http://www.informatizate.net/articulos/webservices_20021227.html