Instituto Politécnico Nacional

Centro de Investigación en Computación

INTERFAZ PARA LA PUBLICACIÓN DE LAS OPERACIONES DE CONSULTA EN UN SISTEMA ADMINISTRADOR DE BASE DE DATOS (SABD) COMO SERVICIOS WEB

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

EL ING. LUIS GONZÁLEZ ESCALANTE

DIRECTOR DE TESIS: **M. en C. ALEJANDRO BOTELLO CASTILLO**

MÉXICO, D.F. **2005**

CGPI-14

INSTITUTO POLITECNICO NACIONAL

COORDINACION GENERAL DE POSGRADO E INVESTIGACION

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 15:00 horas del día 5 del mes de Noviembre de 2004 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"INTERFAZ PARA LA PUBLICACIÓN DE LAS OPERACIONES DE CONSULTA EN UN SISTEMA ADMINISTRADOR DE BASES DE DATOS (SABD) COMO SERVICIOS WEB

GONZÁLEZ	**ESCALANTE**	**LUIS**
Apellido paterno	materno	nombre(s)

Con registro:

B	0	2	0	8	9	1

aspirante al grado de: ***MAESTRO EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

DR. ADOLFO GUZMÁN ARENAS

Secretario

DR. AGUSTÍN FRANCISCO GUTIÉRREZ TORNÉS

INSTITUTO POLITECNICO NACIONAL
COORDINACION GENERAL DE POSGRADO E INVESTIGACION
ENE 7 2005
D. S. E.
COORDINACION DE POSGRADO

Primer vocal

M. EN C. ALEJANDRO BOTELLO CASTILLO

Segundo vocal

M. EN C. GILBERTO LORENZO MARTÍNEZ LUNA

Tercer vocal

M. EN C. SANDRA DINORA ORANTES JIMÉNEZ

Suplente

M. EN C. LEANDRO BALLADARES OCAÑA

EL PRESIDENTE DEL COLEGIO

DR. JUAN LUIS DÍAZ DE LEÓN SANTIAGO

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

Dedicatoria

La realización de este trabajo de investigación, representa un paso mas en el objetivo de mi vida personal y profesional, es por esto que lo dedico a Dios por darme la oportunidad de realizarlo y acompañarme durante el mismo; a mi familia por el apoyo incondicional recibido durante la realización de esta etapa de mi vida; y a mis padres por que sin su apoyo y esfuerzo no hubiera sido posible alcanzarlo.

Luis González

Agradecimientos

A Dios por enseñarme el camino para lograr cada una de mis metas.

A mis padres Luis González Galicia y Magdalena Escalante Rueda por el apoyo incondicional que me han dado a lo largo de mi vida.

A mi familia pero en especial a Patricia, Luis y Karen por el apoyo y esfuerzo que realizaron junto conmigo para lograr juntos este objetivo.

A mi tío Carlos González por darme un lugar donde llegar para realizar este sueño.

A mis profesores que me brindaron los conocimientos y que depositaron en mi una inquietud para realizar este trabajo.

A mis amigos, que por medio de las discusiones y preguntas, me hacen crecer en conocimiento.

A todos los compañeros que he tenido en los lugares que he laborado por su sana competencia e intercambio de conocimiento.

A todos aquellos que de una u otra forma formaron parte de este proyecto, tanto en apoyo moral como en apoyo profesional.

Contenido

Índice de Figuras

Resumen

La necesidad actual de las organizaciones por compartir información a través de un medio de comunicación como Internet, ha generado que algunas empresas líderes en la construcción de Sistemas Administradores de Base de Datos (SABD) inviertan esfuerzos en el desarrollo de nuevas tecnologías que utilicen los estándares que maneja actualmente la WWW (World Wide Web, Telaraña Mundial de la Información), entre los que se encuentran los Servicios Web para el desarrollo de aplicaciones distribuidas y XML (eXtensible Markup Language, Lenguaje Extensible de Marcas) para el intercambio de información.

Por otro lado, las organizaciones manejan componentes de software escritos para diferentes plataformas, los cuales pueden comunicarse a través de una red local (dentro de una misma organización) o en una red de área amplia (entre organizaciones), lo que ocasiona que se necesiten diferentes configuraciones entre aplicaciones de acuerdo a los requerimientos de cada componente.

El propósito general de este trabajo es el desarrollo de una herramienta integrada y multiplataforma que genere Servicios Web para facilitar la publicación de las operaciones de consulta (sentencias SQL (Structured Query Language, Lenguaje Estructurado de de Consulta), procedimientos almacenados y vistas) a los datos contenidos en un SABD y reducir el tiempo de procesamiento y adaptación del usuario.

Abstract

The current needs of organizations to share information using a communication media like the Internet, has generated that top companies in the Database Management System (DBMS) invest their efforts in the development of new technologies that use WWW (World Wide Web) standards like Web Services for distributed applications development and XML protocol for data exchange.

In the other hand, organizations work with software componentes developed for different plataforms, which can communicate within a local network (inside one organization) or in a wide area network (among organizations), increasing needs for different configurations among applications according to each component requirements.

The general intention of this work is to develop an integrated and multiplatform tool that generates Web Services to facilitate the publication of consulting operation results (SQL sentences, stored procedures and views) over a DBMS, reducing processing time and user adaptation.

Capítulo I
Introducción

Actualmente los datos se consideran uno de los recursos más importantes para las organizaciones en cualquier área, no sólo para contar de manera oportuna con ellos, si no para compartirlos con organizaciones afines [L-1]. Sin embargo, al encontrarse contenidos en una diversidad de fuentes de origen, formatos de presentación y de existir diversas tecnologías para almacenarlos, resulta complicado compartirlos.

Este trabajo de tesis presenta una alternativa que utiliza tecnologías como XML (eXtensible Markup Language, Lenguaje Extensible de Marcas), Servicios Web, JDBC (Java Database Conectivity, Conectividad para Bases de Datos en Java) y Java, para permitir al administrador de base de datos publicar de manera simple las operaciones de consulta necesarias para acceder a los datos contenidos en algún SABD (Sistema Administrador de Base de Datos).

Por otro lado, se propone una arquitectura para la atención de peticiones de ejecución de consultas para cada Servicio Web, que sea multiplataforma y configurable de acuerdo al SABD empleado.

1.1 Antecedentes

La tendencia hacia el desarrollo de aplicaciones basadas en el Web, ha hecho que la mayoría de las empresas busquen publicar su información a través de este medio, para consultarla y trabajar con ella [31][L-3].

Los Sistemas Administradores de Bases de Datos (SABD) son programas de software que surgen a partir de la necesidad de administrar los datos almacenados, brindando un entorno seguro y confiable, donde el usuario puede añadirlos, consultarlos y modificarlos de maneara independiente, sin necesidad de ejecutar las aplicaciones propietarias de las organizaciones [26][27]. No obstante, es necesario contemplar aspectos como la seguridad, requiriendo autentificarse para poder procesar los datos, además de contar con los conocimientos necesarios para su manejo.

A la fecha se han desarrollado tecnologías como el XML que permiten intercambiar información de manera segura, confiable y sencilla; además de reducir algunas tareas tediosas para el programador, como la validación de los datos o el recorrido de las estructuras, las cuales están descritas en la especificación del estándar XML [L-8].

Debido a que XML es un estándar relativamente reciente [16], hoy en día los SABD están adoptando este formato para utilizarlo en el intercambio de datos, sin embargo aún existen SABD que no lo soportan, por lo que es necesario utilizar herramientas de terceros (middleware), además de tomarse en cuenta para utilizarlas los requerimientos de la plataforma, lenguaje de programación, entre otros [31].

Los Servicios Web es otra tecnología existente, basados en XML, que permiten la ejecución de operaciones sin importar la plataforma, sistema operativo, ni el lenguaje de programación en el cual estén implementados; ya que el programador sólo debe conocer donde se encuentra publicado el servicio, solicitarlo y procesar los resultados obtenidos [31][L-18], facilitando al programador el desarrollo de aplicaciones.

1.2 Planteamiento del Problema y Justificación

Los diversos formatos existentes para publicar información (páginas web, documentos pdf, archivos excel, entre otros), no permiten llevar a cabo un análisis oportuno de su contenido, principalmente por el proceso de adaptación y en algunos casos de limpieza que deben seguir los datos contenidos en ellos, además que la mayoría no son generados de forma dinámica y sólo presentan cierto nivel de detalle para poder procesarlos debido principalmente, a que no existe interoperabilidad entre ellos y que el usuario no siempre cuenta con los permisos necesarios para acceder a los datos de manera directa. Personas analistas de la información deben realizar trabajos de validación y adaptación, ocasionando perdida de tiempo y que, cuando se presenten los resultados del análisis, la información sea obsoleta.

En el mercado existen tecnologías como ODBC, JDBC, OLEdb y ADO, etc., que facilitan la interoperabilidad con diversas fuentes de información [33]. A pesar de ello, deben contemplarse usuarios, códigos de acceso, entre otros, con los que normalmente el usuario no cuenta. Se propone fortalecer la interoperabilidad publicando los datos contenidos en los SABD con Servicios Web y como formato a XML.

No obstante, para la publicación de Servicios Web es necesario contar con un servidor Web que permita la comunicación en una Intranet y/o Internet, además de las API (Application Programing Interface, Interfaz para programas de aplicación) necesarias para el manejo de mensajes SOAP (Simple Object Access Protocol, Protocolo Simple de Acceso a Objetos), utilizados para comunicar al cliente con los Servicios Web y un archivo WSDL (Web Service Description Language, Lenguaje de Descripción de Servicio Web) que los describe [7][9][L-18].

Este trabajo plantea utilizar los Servicios Web como modelo para publicar las operaciones de consulta a la información almacenada de manera dinámica, eficiente y que pueda ser procesado de manera sencilla (basado en XML), es decir, que el analista pueda obtener los datos que encuentra en la red pero en un formato dinámico que le facilite su trabajo.

1.3 Objetivos

El desarrollo de este trabajo de tesis tiene los siguientes objetivos:

1.3.1 Objetivo General

Desarrollar una herramienta multiplataforma, para crear, administrar y publicar Servicios Web basados en las operaciones de consulta de un SABD y que habilite, la interoperabilidad de la información a través de Internet y/o Intranet.

1.3.2 Objetivos Específicos

- Crear un Servidor HTTP (Hypertext Transfer Protocol, Protocolo de Transferencia por Hipertexto) que se encargue de la publicación de los Servicios Web para bases de datos, permitiendo al cliente la comunicación y transmisión de los datos obtenidos de las consultas, así como interactuar con los mismos.
- Brindar una interfaz gráfica que permita construir y publicar, de manera simple, Servicios Web basados en las funciones de consulta nativas de los SABD (local o remoto) y que son determinadas y escritas por el administrador de la base de datos o por el programador de sistemas.
- Utilizar XML como formato para el envío de resultados generados por la ejecución de las operaciones de cada Servicio Web y fortalecer la interoperabilidad en el intercambio de información, así como guardar la información de configuración de la aplicación.

1.4 Beneficios Esperados

Proporcionar un marco de trabajo que permita a los administradores de base de datos publicar los datos contenidos en un SABD y que actualmente, se encuentran en formatos heterogéneos, cubriendo la necesidad de invertir un tiempo importante en tareas como las siguientes:

- Determinación de la herramienta a utilizar para publicar Servicios Web, de acuerdo a la plataforma software-hardware.
- Instalación y configuración de una herramienta para la publicación de Servicios Web.
- Determinación de los controladores a utilizar de acuerdo al SABD elegido y del lenguaje de programación soportado por el proveedor de Servicios Web.
- Desarrollo de Servicios Web en el lenguaje de programación elegido.

Por otro lado, la mayoría de estas aplicaciones pueden necesitar licencias para poder utilizarlas.

Se busca que esta herramienta ayude a lograr una mayor interoperabilidad entre las diferentes fuentes de información, ya que utiliza un estándar de intercambio de información como XML, al que pueden aplicarse plantillas de visualización y/o transformación, permitiendo al usuario concentrar en un sólo documento datos de diferentes fuentes.

1.5 Alcances y Límites

Este trabajo integra la generación y publicación de Servicios Web para la ejecución de las operaciones de consulta de un SABD. Las operaciones que se cubren son la ejecución de procedimientos almacenados (donde aplique), consulta de vistas y la ejecución de sentencias SQL (Structured Query Language, Lenguaje de Consulta Estructurado) dadas por el administrador de la base de datos; éstas pueden estar contenidas en el SABD o ser definidas en forma dinámica.

La herramienta desarrollada cuenta con módulos que permiten al usuario construir sentencias SQL, las cuales, de acuerdo con las características del SABD, pueden ser almacenadas como procedimientos almacenados y/o vistas para su posterior publicación o sentencias que serán publicadas como Servicio Web en forma directa.

Esta arquitectura cumple con las necesidades para la publicación de Servicios Web, como son: un Servidor HTTP y las funciones de manipulación y generación de mensajes SOAP. Por otro lado, también se brinda interacción directa con cualquier SABD, que soporte preferentemente, la ejecución de Procedimientos Almacenados y que tenga implementado un controlador JDBC para su conexión. Adicionalmente, la información arrojada por las operaciones de consulta publicadas como Servicio Web son enviadas como documentos XML.

La organizción del documento esta determinada por el camino que siguió la investigación, presentando en el capítulo dos el marco teoríco sobre algunas de las tecnologías que

llevaron a realizar este trabajo; el capítulo tres presenta el análisis y diseño de la aplicación; el capítulo cuatro contiene la implementación de la aplicación; el capítulo cinco describe las pruebas realizadas y algunos comparativos con herramientas semejantes; por último el capítulo seis, muestra las conclusiones generadas durante el desarrollo de este trabajo así como algunos trabajos futuros que pueden realizarse al mismo.

Capítulo II
Marco Teórico

El trabajo esta orientado a brindar una alternativa que ayude a resolver el problema de interoperabilidad entre diversas fuentes de información, publicando los datos contenidos en los SABD (Sistemas Administradores de Base de Datos) y facilitando el intercambio de los mismos entre aplicaciones heterogéneas a cualquier nivel de desarrollo, utilizando tecnologías como XML y Servicios Web, estos últimos (durante el desarrollo de este trabajo) aún no se enfocaban a la ejecución de sentencias SQL(Structured Query Language, Lenguaje Estructurado de Consultas) en una Base de Datos.

2.1 Panorama General

Las bases de datos han ido de la mano con la evolución del desarrollo de sistemas de información, al mismo tiempo han surgido problemas como la integración y el intercambio de información. Al existir diversas fuentes de información, tales como los archivos planos, las hojas de cálculo y documentos independientes, se dificulta la integración y la consulta de estas fuentes [L-45].

Peter C. Lockemann [L-46] menciona que las tecnologías referentes a las comunicaciones y al manejo de información han crecido de manera importante en los últimos años y brindan, la posibilidad de compartir información entre sistemas de información internos y externos de las organizaciones; sin embargo, los sistemas existentes están desarrollados, en su mayoría, en un ambiente distribuido, utilizando diferentes plataformas interconectadas entre sí a través de una Intranet o Internet. Muchos de estos sistemas son autónomos y la información que contienen se encuentra almacenada en bases de datos heterogéneas, las cuales pueden estar manipuladas por SABD de distinta naturaleza [L-47].

Hoy en día Internet es el medio más utilizado para intercambiar y publicar información. En [L-41] se menciona que fue el principal detonador para compartir datos entre sistemas distribuidos heterogéneos, sin embargo, para consultar información se utiliza la interfaz de las herramientas para navegar por dicho medio. A pesar de permitir la consulta de información de fuentes diferentes, la publicación de los datos es realizada a través de HTML (HyperText Markup Language, Lenguaje de Marcación de Hipertexto), lo que lleva a que la interpretación de la información por parte de las aplicaciones (no de personas) sea pobre y costosa, debido a las necesidades de desarrollo que se deben considerar y a la poca certidumbre que existe sobre los datos consultados por estas aplicaciones.

Como se observa, la integración de la información y el intercambio de información entre sistemas presentan un problema de interoperabilidad, debido a las características mismas de los actores que interactúan, por lo que se han realizado diversos esfuerzos para ayudar a dar solución a estos problemas.

Cuando surgen los SABD se pensaba que el desarrollo de aplicaciones con acceso a base de datos sería simple y ayudaría a disminuir el tiempo de desarrollo de sistemas de información, al igual que la integración de los datos contenidos en diferentes SABD, sin embargo a lo largo de su historia, han surgido diversas compañías que se han encargado de construir SABD que, a pesar de la existencia de estándares como SQL-92 y SQL-99, no son soportados de manera completa por estas herramientas, por lo que el problema de interoperabilidad es en el momento de solicitar la ejecución de sentencias SQL en SABD distintos; por otro lado, la diversidad de agregados que cada compañía inserta en su SABD hacen que este problema crezca y dificulte la integración de información entre diversas bases de datos.

2.1.1 Interoperabilidad

El problema presentado en la sección 2.1 puede resumirse como la falta de interoperabilidad entre los sistemas internos y entre sistemas externos a una organización, no sólo entre aplicaciones que utilizan las personas, sino también puede presentarse a nivel de la base de datos, como en los nombres y tipos de los atributos, en los esquemas y hasta en el modelo de datos, por mencionar algunos. Por otro lado, existen otros niveles donde se puede presentar este problema [L-48].

Un concepto de interoperabilidad [L-49] es: "la habilidad de los sistemas para intercambiar información e interpretarla de manera correcta", pero para poder cumplir con esta definición, es necesario adaptar los sistemas existentes a nuevas interfaces.

Cuando las organizaciones son dueñas de los sistemas que necesitan interactuar y ser interoperables, la tarea es sencilla, porque se tiene el conocimiento de la estructura de los sistemas, aunque el grado de complejidad dependerá de la cantidad de sistemas heredados que en dicha organización son operados. Por otra parte, cuando la integración de información se realiza a través de la prestación de los datos consultados en Internet y obtenidos como páginas Web, se vuelve una tarea compleja. Es por ello, que hoy en día muchas organizaciones invierten alrededor del 30% del presupuesto destinado a tecnología de la información, en la integración de sus sistemas de información y en la integración de sistemas externos a la compañía debido a la evolución de la misma o a la fusión con otras compañías.

Existen tecnologías que permiten lograr una interoperabilidad entre fuentes heterogéneas de información, permitiendo a las aplicaciones su interpretación y un procesamiento correcto, las cuales son descritas en [L-50]; sin embargo, estas tecnologías están orientadas a ser aplicadas en donde los encargados de su implementación tienen acceso a las fuentes heterogéneas y conocen los esquemas de las bases de datos, así como la estructura interna de los sistemas, adicionalmente que el costo de implementación puede ser elevado.

Otro tipo de tecnologías han surgido para ayudar a resolver este problema, sin la necesidad de conocer la estructura de los sistemas con los cuales se desea intercambiar información [L-49]; pese a ello aún existen requerimientos específicos para poderlas implementar.

2.2 Integración de Información

El proceso de integración puede definirse como la combinación e interacción de datos y funciones de un sistema con otros y para llevarlo a cabo, es necesario tener conocimiento de las relaciones entre datos y procesos. A. Zisman menciona [L-50] que un gran número de metodologías, herramientas y soluciones para integrar bases de datos, han sido propuestas y presentadas a nivel de estudio, pero también han existido un número importante de soluciones a nivel comercial. Sin embargo a pesar de la existencia de estos algoritmos, herramientas y metodologías que ayudan al proceso de integración, aún es necesario la intervención del usuario para la resolución de conflictos. Por otro lado, el crecimiento de la infraestructura de las empresas, la evolución de la tecnología de comunicación y el surgimiento de un medio de comunicación como Internet, ha llevado a adoptar una plataforma de desarrollo común para la mayoría de las organizaciones que buscan publicar su información a través de un medio público y masivo; en donde, la existencia de un inmenso volumen de información y que las personas sean parte de este proceso de integración, aumenta la posibilidad de errores presentados en el mismo proceso.

Por consiguiente, el proceso de integración de información se vuelve un paso tedioso al realizar tareas como:

- Localización y selección de la información deseada.
- Identificación del formato en el que se presenta la información.
- Limpieza de la información encontrada (si es que se encuentra).
- Adaptación al nivel de detalle deseado (en caso de que la información así lo permita).
- Integración de la información procesada en el formato deseado.
- A lo largo de este proceso se pueden presentar errores, principalmente en aquellos que dependan de los usuarios, tal como se muestra en las figuras 2.1 y 2.2.

El problema de la interoperabilidad al que se enfoca este trabajo es a nivel de la publicación de los datos contenidos en los SABD en un formato estándar y utilizando tecnologías independientes de la plataforma de desarrollo, buscando lograr que la interacción con las fuentes de información a través de un medio de comunicación se realicen de forma factible como se muestra en la figura 2.3, minimizando la intervención humana para este proceso de integración.

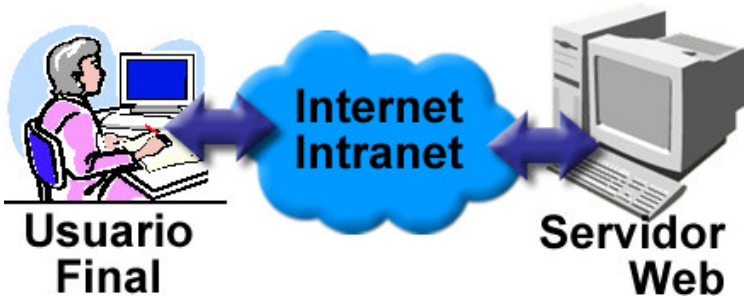


Figura 2.1 Interacción con Aplicaciones para Web.

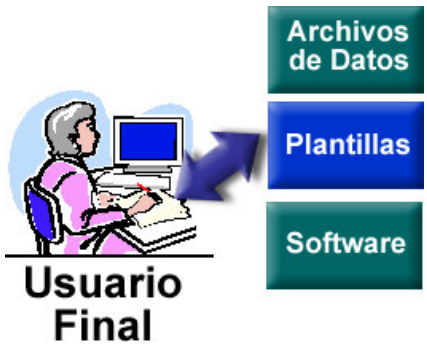


Figura 2.2 Interacción tradicional entre el Usuario y las Aplicaciones

Revisando las tecnologías desarrolladas para resolver el problema de interoperabilidad, actualmente XML (eXtensible Markup Language, Lenguaje Extensible de Marcas) ha sido aceptado como un estándar para el intercambio de información, además se han desarrollado otras tecnologías entorno él, una de ellas, son los Servicios Web que también han tenido una aceptación importante para interactuar entre aplicaciones.

Figura 2.3 Interacción entre Aplicaciones

2.3 Servicios Web

De forma natural los Servicios Web han surgido como parte de la evolución de internet [L-44]. Por otro lado, los programación basada en componentes ha llegado a ser popular y hoy en día, es difícil construir una aplicación que no este influenciada por ellos.

2.3.1 Definición

En [L-44] se define el término Servicios Web como: una aplicación de red habilitada para Internet utilizando protocolos estándares de comunicación entre aplicaciones, a través de interfaces bien definidas y que son descritas utilizando un lenguaje estándar. Otra definición tomada de [L-43] dice que un Servicio Web puede ser definido como una aplicación modular, generalmente independiente y que se describe así misma, localizada e invocada a través de Internet o una Intranet. Existen diversas definiciones para este tipo de tecnología, que van de acuerdo a la funcionalidad que brindan. Una definición general de lo que son los Servicios Web es: aplicaciones XML que son descritas, publicadas, invocadas y localizadas a través de un medio de comunicación, principalmente Internet, que facilitan el acceso a aplicaciones, objetos, base de datos o aplicaciones de comercio electrónico [1][11].

Los Servicios Web ayudan en la integración de aplicaciones y lo hacen con un bajo costo a comparación de otras tecnologías. Representan una nueva forma de aplicación intermediaria basada en XML y principalmente en Internet. XML e Internet han ayudado en la solución de los retos que son asociados con la interacción entre aplicaciones [9].

2.3.2 Beneficios

Los Servicios Web brindan una manera estándar de conectividad e integración entre sistemas con diferentes formatos de operación y que pueden estar localizados en diferentes plataformas, además de poder ser invocados a través de protocolos de comunicación, principalmente HTTP [9][11].

Entre los beneficios que esta tecnología nos proporciona encontramos los siguientes:

- **Universales:** están basados en el formato XML, el cual ya es un estándar y permite definir de manera sencilla la utilización de las operaciones en cada Servicio Web, además de saber que el tipo de información que intercambiará el servicio con la aplicación cliente, estará en base a los estándares de éste formato.
- **Modularidad:** pueden describir, publicar e invocar aplicaciones enteras o parte de ellas, permite encapsularlas y dejar sólo a la vista la descripción de las operaciones permitidas, ocultando la implementación.

- **Interoperabilidad en Negocios:** al ser interoperables, los Servicios Web permiten que la asociación entre empresas se pueda realizar de manera dinámica y automática, además de facilitar el intercambio de información entre empresas de diferentes áreas.
- **Accesibilidad:** al poder ser publicados por Internet, podrán ser accedidos por una gran variedad de dispositivos de comunicación.
- **Integración de Sistemas Heredados:** facilitan la interacción de las aplicaciones actuales con los sistemas heredados, ya que las aplicaciones harán uso de los Servicios Web para comunicarse con este tipo de sistemas, sin necesidad de preocuparse la forma en que está implementado el sistema heredado.
- **HTTP:** El utilizar los protocolos de Internet para poder comunicarse con los Servicios Web, hace que puedan pasar a través de los proxies y firewalls en los que otras tecnologías son bloqueadas.
- **Independencia de Lenguaje y Plataforma:** Los Servicios Web pueden ser desarrollados en un lenguaje de programación y ser utilizados por otro, además que no dependen de la plataforma en que se desarrollen.

Sin embargo, existen diferentes áreas para las cuales se han desarrollado herramientas para la construcción de Servicios Web. La construcción de Servicios Web orientados a base de datos cuenta con pocas herramientas, lo que es uno de los objetivos del desarrollo de este proyecto, que aprovecha las tecnologías antes mencionadas.

Existen en el mercado un gran número de aplicaciones que permiten construir y publicar Servicios Web (Axis, MS SOAP Toolkit, Apache SOAP, Oracle, entre otros) [L-25][L-26][L-27][L-28][L-29][L-30][L-31], sin embargo el uso de las mismas puede complicarse ya que es necesario:

- Saber un lenguaje de Desarrollo.
- Contar con un Servidor HTTP para poder publicar Servicios.
- Tener conocimientos para configurar el servidor.
- Contar con herramientas de terceros para su instalación.
- Contar con la plataforma adecuada.
- Si se desea acceder a base de datos se debe contar con los controladores de acuerdo a la plataforma de desarrollo.
- Conocer la forma en que debe generarse la descripción del Servicio.

Esta aplicación busca ofrecer un marco de trabajo para desarrollar Servicios Web con un propósito específico (base de datos) sin la necesidad de conocer un lenguaje de desarrollo e independiente de la plataforma y que genere de manera automática la descripción de los mismos.

2.3.3 Arquitectura

La arquitectura básica de los Servicios Web proporciona una interacción entre diferentes componentes y tecnologías [1][11][L-18]. La figura 2.4 muestra su arquitectura básica.

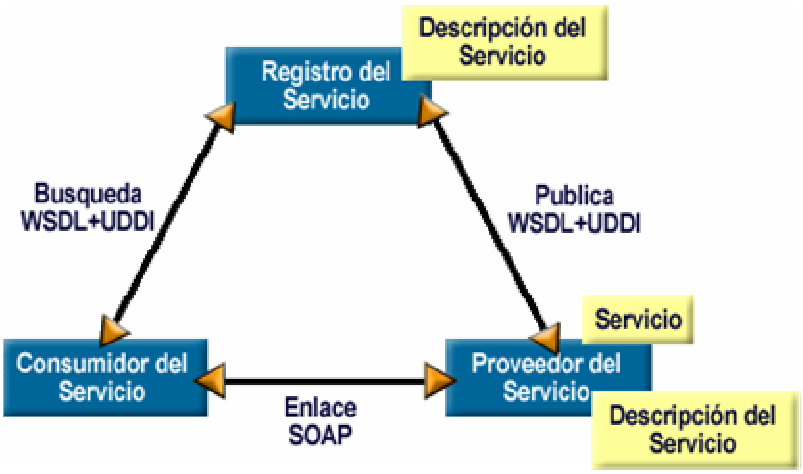


Figura 2.4 Arquitectura Básica de los Servicios Web

En ésta arquitectura se definen los siguientes elementos:

- **Proveedor del Servicio:** es el encargado de proporcionar y atender las peticiones de clientes que soliciten la ejecución de las operaciones de un Servicio Web.
- **Solicitante del Servicio:** son las aplicaciones que solicitan la ejecución de un Servicio Web. Estas aplicaciones pueden ser locales o remotas sin importar la plataforma de desarrollo de las mismas.
- **Registro de los Servicios:** permite la publicación de los servicios desarrollados, además de contar con un mecanismo de búsqueda de servicios proporcionados por otras compañías.

Los componentes clave para los Servicios Web son [7]:

- **WSDL (Web Service Description Language):** describen la forma en que están conformados los Servicios Web.
- **SOAP (Simple Object Access Protocol):** protocolo para intercambiar información en un ambiente distribuido.
- **UDDI (Universal Description, Discovery and Integration):** es un directorio donde se pueden publicar y buscar Servicios Web.

Las secciones siguientes describen cada uno de los componentes mencionados.

2.3.3.1 SOAP (Simple Object Access Protocol)

Es un protocolo basado en XML para mensajería y comunicaciones estilo RPC para comunicaciones entre aplicaciones (en un ambiente distribuido) [10][L-7][L-12].

Brinda mecanismos para la ejecución de llamadas a procedimientos remotos entre programas, de tal manera que se puedan establecer de manera eficiente las comunicaciones entre aplicaciones.

La estructura de un mensaje SOAP se presenta en la figura 2.5.

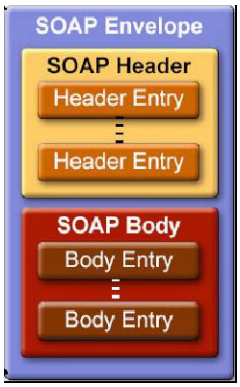


Figura 2.5 Estructura de Mensaje SOAP

Como se observa, un mensaje SOAP está compuesto por un contenedor (envelope) que a su vez contiene el cuerpo y el encabezado del mensaje. A continuación se muestra un ejemplo de mensaje SOAP.

Mensaje SOAP

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:getRate xmlns:m="urn:xmethods-CurrencyExchange">
            <country1 xsi:type="xsd:string">String</country1>
            <country2 xsi:type="xsd:string">String</country2>
        </m:getRate>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.3.3.2 WSDL (Web Service Description Language)

WSDL es un lenguaje basado en XML que brinda la información necesaria al cliente para interactuar con el Servicio Web [7][11][L-11]. WSDL es extensible y puede ser usado para describir prácticamente cualquier servicio de red, incluyendo SOAP sobre HTTP.

El elemento raíz de un archivo WSDL es el elemento *definitions*, del cual descienden cinco tipos de elementos:

- **Types:** contiene la definición del esquema de los mensajes que pueden ser enviados y recibidos por el dicho mensaje.
- **Message:** sirve como una referencia entre el mensaje y su definición dentro del esquema.
- **portType:** define un conjunto de interfaces con las que el Servicio Web cuenta. Cada interfaz puede asociarse a más de un mensaje.
- **Binding:** asocia la definición del elemento *portType* con un protocolo en particular.
- **Service:** define una colección de endpoints (puntos de entrada) proporcionados por el Servicio Web.

Cada elemento tiene una definición detallada. Para consultar la sintaxis completa de un documento WSDL refiérase al anexo B.

2.3.4 Servicios Web y SABD

Sin importar el camino que tome el desarrollo de software en los siguientes 10 años, las bases de datos seguirán siendo las principales herramientas para el manejo de los datos [15][27][L-2].

Problemas de interoperabilidad e integración surgen cuando las empresas buscan construir sus aplicaciones en base a lo nuevo o con una sola plataforma. Es por ello que tecnologías como XML, mensajería XML, Servicios Web o XML databases han comenzado a tener más aceptación [15][22].

Actualmente los principales productores de herramientas para la administración de datos, han orientado sus esfuerzos a incorporar el soporte necesario para las tecnologías antes mencionadas. Oracle, MS-SQL Server, IBM DB2, entre otros, han incluido algunas capacidades para poder manejar XML, mensajería XML, Servicios Web y XML databases. Sin embargo, las capacidades actuales de los SABD de éstos fabricantes aún no permiten, de manera sencilla, configurar Servicios Web directamente para sus bases de datos, ya que se necesita recurrir a herramientas propietarias nuevas (proporcionadas por los fabricantes de SABD) o desarrollar código extra para una aplicación que habilite consultas a las bases de datos como Servicios Web.

Cabe mencionar que estos SABD son comerciales, y por lo que respecta a los SABD libres, la incorporación de tecnologías como XML ha sido más lenta, e incluso no soportan algunas operaciones de consulta como procedimientos almacenados; en estos casos, es necesario recurrir a buscar herramientas o desarrollar las propias para poder soportar XML.

2.3.5 Herramientas con Soporte para Servicios Web

El común para las herramientas existentes para publicar Servicios Web (Microsoft .NET[L-25], Java Web Services Development Pack [L-26], WebSphere[L-27], Apache SOAP [L-28], Axis[L-29], JDeveloper[L-30]), es que están realizadas para una plataforma de desarrollo específica, por lo que los desarrolladores que desean publicar servicios deberán elegirla de acuerdo a sus posibilidades y capacidades.

Otra característica común ,es que son pensadas para propósitos generales y aún se deben seguir desarrollando módulos externos para ofrecer el intercambio de información en formatos estándares como XML.

El caso particular de *Virtuoso*[L-31], herramienta semejante a la desarrollada en esta trabajo, presenta una orientación específica a la publicación de Servicios Web para bases de datos, sin embargo es una herramienta comercial y para utilizarla debe pagarse una licencia.

2.4 Resumen

Este capítulo presentó un panorama general de las tecnologías utilizadas para el desarrollo de este trabajo, además de mostrar la importancia que actualmente tiene no contar con herramientas adecuadas para el intercambio de datos entre organizaciones.

Capítulo III
Análisis y Diseño

El análisis y diseño de la arquitectura que comprende la aplicación desarrollada fueron realizados mediante el uso del UML (Unified Model Language, Lenguaje Unificado de Modelado).

3.1 Requerimientos

Para cumplir con estas orientaciones deben cubrirse los siguientes requisitos para el desarrollo de la aplicación:

- Contar con una interfaz gráfica simple que permita al usuario de manera visual configurar todos los elementos de la aplicación.
- Permitir al administrador de las base de datos definir las conexiones a un SABD y contar con un módulo para interactuar con ellas.
- Cubrir las necesidades para la construcción y publicación de Servicios Web, como son la descripción del Servicio Web (WSDL), además de interpretar y generar mensajes SOAP (Simple Object Access Protocol, Protocolo de Acceso Simple a Objetos).
- Brindar un servidor HTTP (HyperText Transfer Protocol, Protocolo de Transferencia de Hipertexto) para la publicación de los Servicios Web generados en la aplicación.
- Configurar los Servicios Web en base a los elementos contenidos en el SABD, en donde el cuerpo de las operaciones por servicio contendrán sentencias SQL para interactuar con el SABD elegido.
- Generar los resultados de la ejecución de las operaciones de Servicios Web en XML.

3.2 Arquitectura General

En la arquitectura mostrada en la figura 3.1, se observan las dos partes principales de la aplicación.

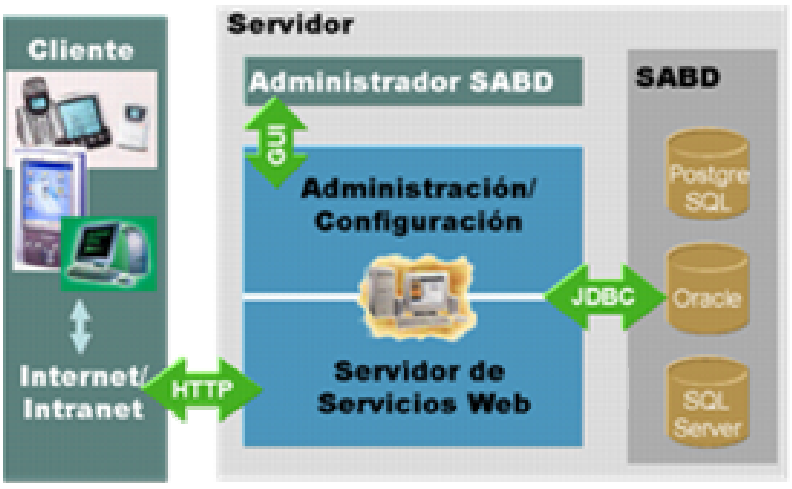


Figura 3.1 Arquitectura General

De manera general los clientes se conectan con una aplicación (basada en el Web o autónoma), la cual solicita la ejecución de los Servicios Web al Servidor de Servicios Web; éste se encarga de validar la solicitud y procesar la petición del servicio, interactuar con el SABD y retornar los resultados en formato XML (esta parte se describe en la sección 3.2.1).

Por otro lado, existe el administrador del SABD quién emplea la parte de Administración/Configuración (descrita en la sección 3.2.2), para publicar los procedimientos almacenados como Servicios Web; presenta una interfaz gráfica y simple, e interactúa con el SABD a través del controlador JDBC seleccionado por el administrador.

3.2.1 Servidor de Servicios Web

Esta parte de la aplicación integra los elementos necesarios para la publicación de Servicios Web, en donde la arquitectura propuesta para éste módulo es la siguiente (figura 3.2):

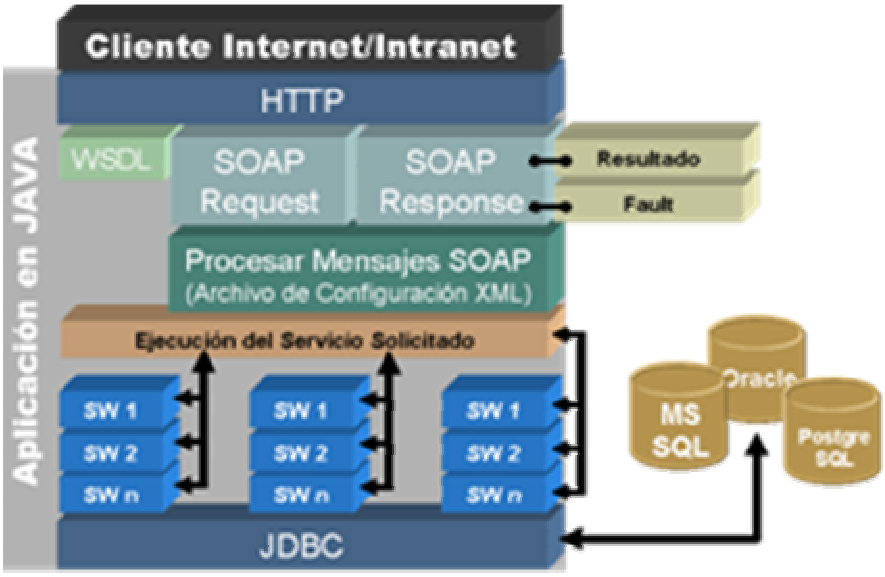


Figura 3.2 Arquitectura Servidor de Servicios Web

Los elementos de los cuales se compone son lo siguientes:

- **Servidor HTTP:** ofrece la posibilidad de solicitar una autentificación al usuario de éste servidor, que debe estar dado de alta en la configuración del mismo, además se encarga de identificar y atender uno de las siguientes mensajes:
    - **Solicitud del WSDL:** para que el cliente de los Servicios Web pueda utilizarlos, se debe solicitar al Servidor de Servicios Web el archivo WSDL el

cual describe las características del servicio, así como las operaciones que se pueden realizar mediante el uso de una URL como se muestra en ejemplo:

http://quiahuitl.cic.ipn.mx:8080/SWSistNomina.wsdl

- **SOAP Request:** es el mensaje de requerimiento de ejecución de Servicios Web, el cual contiene la información necesaria para poder ejecutarlo, como se muestra en la figura 3.3.
- **SOAP Response:** contiene el mensaje SOAP con los resultados arrojados por la ejecución del Servicio Web en formato XML. En caso de que el mensaje SOAP recibido desde el cliente sea incorrecto o exista un error durante el proceso de ejecución del Servicio Web, se deberá enviar un mensaje SOAP (de tipo fault) indicando el tipo de error que se encontró y una descripción del mismo, puede observarse en la figura 3.4.

Mensaje SOAP

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <m:GetCatDep xmlns:m="urn:WSSABD:ServicioOracle" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 3.3 Mensaje SOAP de Petición de Servicio Web

- **Procesar Mensajes SOAP:** son las funciones necesarias para validar los mensajes recibidos y generar el mensaje que se envía al cliente que solicitó la ejecución del servicio. La validación del servicio no sólo consiste en la estructura del mensaje SOAP, sino que además debe estar ligado con un archivo de configuración en XML generado por el administrador del SABD.
- **Ejecución del Servicio:** una vez que se cumplieron con los requisitos para la ejecución del servicio, se ejecuta la operación solicitada y se interactúa con el SABD; cuando la ejecución concluye, se forma el documento XML con los resultados o en su caso, el error generado por el proceso para enviar el mensaje de respuesta.

Es importante señalar que esta aplicación fue diseñada para soportar la comunicación con diferentes SABD ubicados en lugares remotos; esto es, que si el usuario de la aplicación desea publicar Servicios Web para uno o más SABD, no será necesario que esta aplicación se instale en cada equipo donde se encuentre el SABD, ya que sólo será necesario que exista una comunicación a través de Internet entre el equipo donde esta conectada ésta aplicación.

Mensaje SOAP de Respuesta

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <todo:INTERPRETADO xmlns:todo="urn:xmethods-
CurrencyExchange">
            <todo:Resultado>&lt;?xml version=&quot;1.0&quot;
encoding=&quot;UTF-8&quot;?&gt;&#xd;
&lt;Resultado_Operacion&gt;&lt;Res_Inst_0&gt;&lt;Registro&gt;&lt;IDDE
PT&gt;&lt;![CDATA[10]]&gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;&lt;![CDATA
[ACCOUNTING]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;Registro&gt;&lt
;IDDEPT&gt;&lt;![CDATA[20]]&gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;&lt;![C
DATA[RESEARCH]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;Registro&g
t;&lt;IDDEPT&gt;&lt;![CDATA[30]]&gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;&lt
;![CDATA[SALES]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;Registro&gt;
&lt;IDDEPT&gt;&lt;![CDATA[40]]&gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;&lt;!
[CDATA[OPERATIONS]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;/Res_I
nst_0&gt;&lt;/Resultado_Operacion&gt;</todo:Resultado>
        </todo:INTERPRETADO>
    </soapenv:Body>
</soapenv:Envelope>
```

Mensaje SOAP con Error

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <soapenv:Fault>
            <faultcode>soapenv:Error en la ejecucion de la operacion
solicitada</faultcode>
            <faultstring>La fuente de Datos para el Servicio:
SERVICIOORACLE no esta disponible por el momento</faultstring>
            <faultactor>ERROR EN SERVIDOR</faultactor>
            <detail/>
        </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

Figura 3.4 Mensaje SOAP de Respuesta y SOAP con Error

3.2.2 Administración/Configuración

La parte de Administración/Configuración se encarga de interactuar con el SABD local o remoto, a través de una interfaz gráfica; la arquitectura para esta parte de la aplicación se muestra en la figura 3.5.

Este módulo se encarga de la construcción de los Servicios Web, así como de la configuración del Servidor HTTP.

Las tareas que cubre esta parte de la aplicación son:

Figura 3.5 Arquitectura para el Módulo Administración/Configuración

- **Configuración del Servidor HTTP:** el proveedor define los parámetros para el funcionamiento del servidor:
    - Punto de entrada para los Servicios Web.
    - Si el servidor web necesitará o no autentificación por parte del consumidor.
    - Puerto en el que se recibirán las solicitudes.
    - Tiempo máximo de espera de las solicitudes.
    - El número máximo de solicitudes que puede atender.

La autentificación de los usuarios en el servidor web se realiza utilizando autentificación básica [40], que en términos generales es una secuencia de texto codificada en Base64, y que es parte de la especificación de HTTP. El siguiente es un ejemplo del mensaje que se envía al consumidor para permitir el acceso al uso de los Servicios Web:

Código General

```
HTTP/1.1 401 Authorization Required
WWW-Authenticate: basic realm="Es necesario un
Usuario y clave de acceso"
```

- **Administración de Controladores JDBC (Conectividad para Bases de Datos en Java):** permite al proveedor indicar a la aplicación la ubicación de los archivos JAR

(Java Archive, Archivo Java) que contienen controladores JDBC de cada SABD los cuales son cargados en el momento que se dan de alta y cuando se inicia la aplicación.

- **Administración Conexiones JDBC:** permite al proveedor configurar las conexiones a los SABD a través de los controladores JDBC tipo 1 y 4. Para poder configurar las conexiones, el proveedor debe indicarle a la aplicación cual será el controlador a utilizar, además de proporcionar los parámetros de conexión (URL de acceso, usuario y clave de acceso) necesarios para comunicarse con el SABD elegido.
- **Administración de Servicios:** el proveedor construye las características generales de los Servicios Web que estarán activos para su publicación. Cada vez que se modifica su configuración deberá generarse la descripción de cada Servicio Web (WSDL). En este módulo, el proveedor determina la conexión JDBC a utilizar así como las operaciones que contendrá cada servicio.
- **Administración de Operaciones:** a través de esta parte de la aplicación el proveedor puede definir las operaciones que contendrá cada Servicio Web, así como el nombre y tipo de cada parámetro de cada operación: Las operaciones podrán tener o no parámetros.
- **Creación del Cuerpo por Operación:** se muestra la información sobre los elementos disponibles en la base de datos para poder construir el cuerpo de cada operación, permitiendo crear en forma visual las sentencias SQL que serán ejecutadas por cada operación de cada Servicio Web, de acuerdo a la conexión elegida en la configuración del Servicio Web. La información mostrada esta agrupada en base a las sentencias SQL permitidas para publicarse (SELECT, INSERT, UPDATE, DELETE y CALL), además de determinar la relación de los parámetros de operación con las sentencias SQL.

3.3 Modelado de la Aplicación

Una vez determinada la arquitectura general de ésta aplicación y en base a los requerimientos establecidos por las partes involucradas (XML, Servicios Web, JDBC y SABD), a continuación se presenta los diagramas UML desarrollados durante el análisis.

La figura 3.6 muestra el modelado de la aplicación, presentando los principales componentes de la misma, empleando los diagramas de casos de uso.

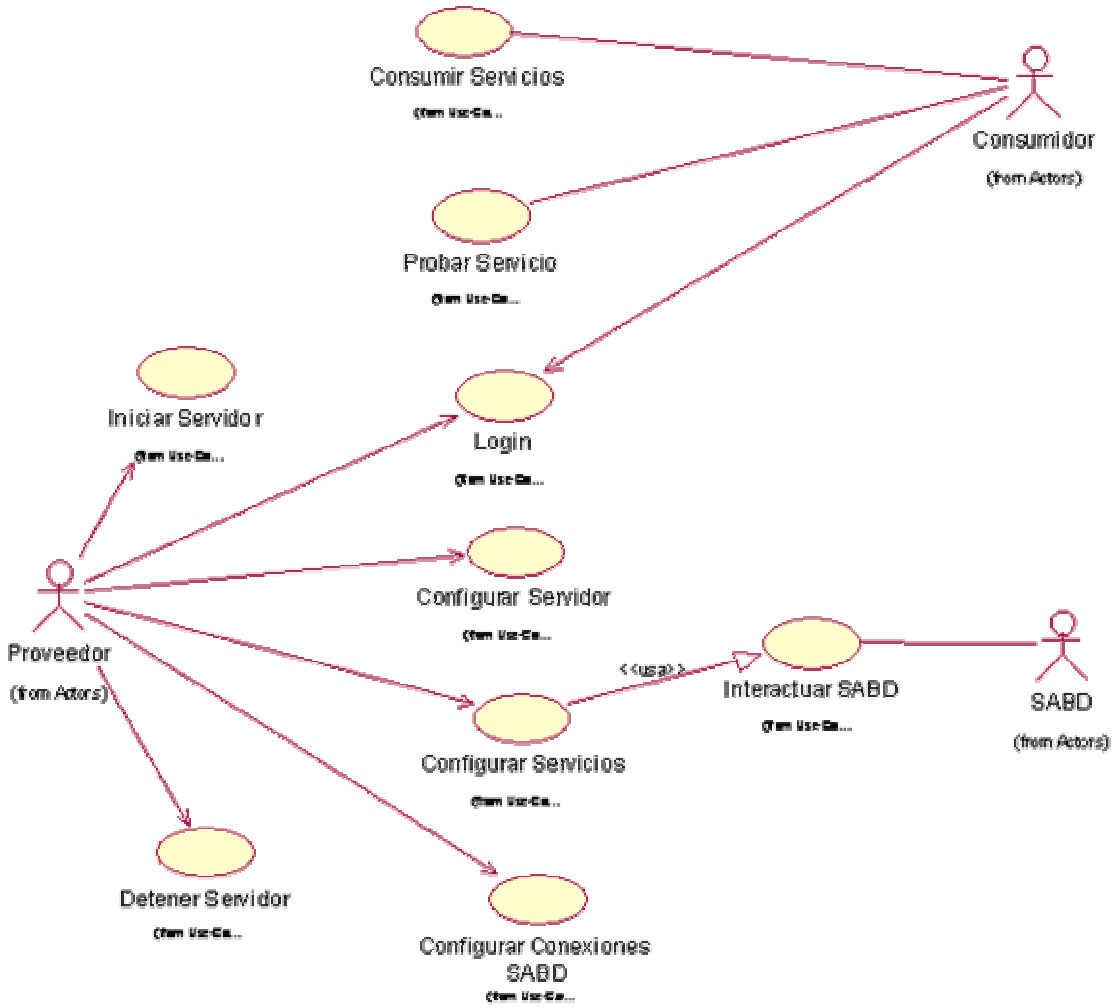


Figura 3.6 Diagrama de Caso de Uso General

Como se observa los principales elementos son los siguientes:

3.3.1 Actores

Proveedor: es el encargado de configurar tanto el servidor de Servicios Web como los propios Servicios Web. Es recomendable que sea el Administrador de la base de datos.

Consumidor: Cualquier aplicación de usuario que solicite la ejecución y descripción de los Servicios Web definidos en esta aplicación.

SABD: Cualquier Sistema Administrador de Base de Datos que cuente con un controlador JDBC tipo 1 o tipo 4.

3.3.2 Casos de Uso

La siguiente es una descripción general de los mismos. La descripción de los principales casos de uso se verá con más detalle en las siguientes secciones:

Configurar Servidor: permite al proveedor de los Servicios Web determinar todos los parámetros necesarios para la operación de esta aplicación.

Iniciar Servidor: el proveedor activa el servidor de Servicios Web y con el se activan las conexiones JDBC a los SABD para poder dar atención a las solicitudes enviadas por los consumidores de servicios.

Detener Servidor: el proveedor concluye el servicio de atención de solicitudes del servidor y con ello se cierran las conexiones a los SABD.

Consumir Servicios: proporciona el medio por el cual los consumidores de Servicios Web solicitan la descripción o ejecución de las operaciones de los servicios. Es un servidor HTTP configurable de acuerdo a los parámetros definidos en el caso de uso Configurar Servidor. En caso de que la solicitud sea obtener el archivo descriptor de un Servicio Web, de acuerdo al URL (Uniform Resource Locator, Localizador Unificado de Recursos) proporcionado, se busca el archivo, y en caso de que no exista se notificará con una página de mensaje de error.

Configurar Servicios: permite al proveedor de los Servicios Web definir los parámetros para cada uno de ellos, así como las operaciones y el cuerpo de cada operación (sentencias SQL).

Interactuar SABD: se encarga de interactuar con los SABD para la creación y atención de Servicios Web, además de que proporciona los elementos y características de la base de datos, así como de los resultados arrojados por la ejecución de las sentencias SQL.

Configurar Conexiones SABD: permite al administrador determinar la configuración de las conexiones que estarán activas para la atención de los Servicios Web. Aquí es donde se actualiza el archivo de configuración XML.

Probar Servicio: brinda al consumidor una interfaz Web para acceder a los Servicios Web configurados y ejecutarlos sin la necesidad de un software especializado.

Login: utilizado por el proveedor y el consumidor para permitir el acceso a la configuración o al consumo de Servicios Web.

3.3.3 Caso de Uso Configurar Servidor

En esta parte de la aplicación, el proveedor de Servicios Web se encarga de definir los parámetros sobre los cuales estará trabajando el servidor HTTP, para atender las solicitudes enviadas a dicho servidor la figura 3.7 muestra el diagrama de colaboración para este caso de uso.

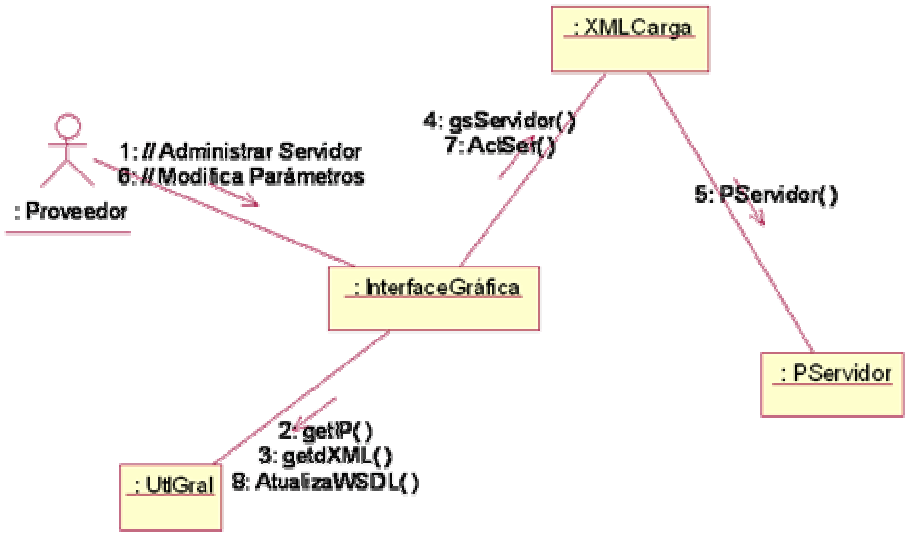


Figura 3.7 Diagrama de Colaboración para Configurar Servidor.

De manera general, el comportamiento es el siguiente:

- El usuario selecciona configurar el servidor HTTP.
- Se recuperan los datos actuales de configuración.
- Se despliega el formato de modificación.
- El usuario actualiza los parámetros.
- Se actualiza el archivo de configuración.

3.3.4 Caso de Uso Iniciar Servidor

El proveedor de Servicios Web debe iniciar el servidor para tener acceso a los Servicios y habilitar las conexiones configuradas a los diferente SABD, la figura 3.8 muestra el diagrama de colaboración para este caso de uso.

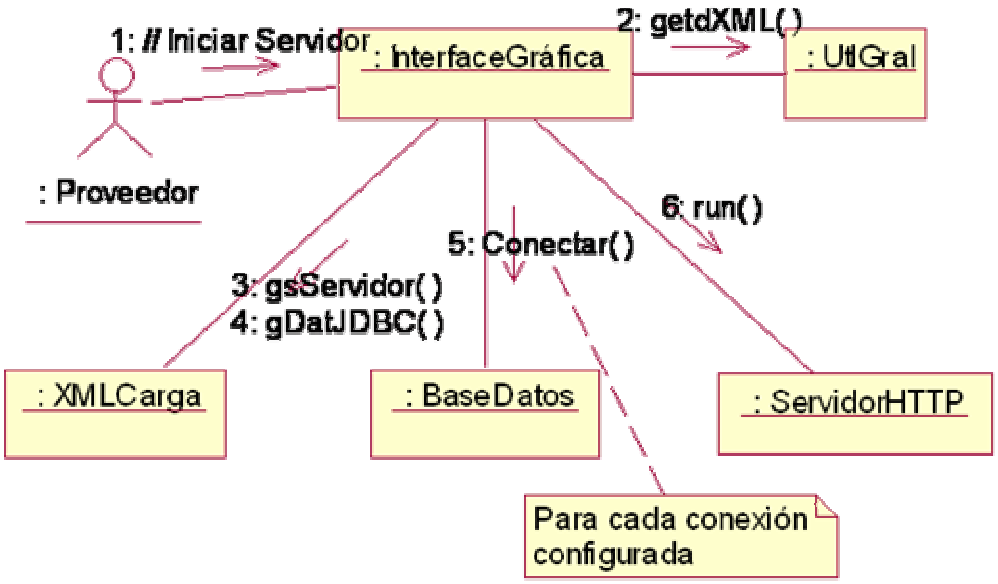


Figura 3.8 Diagrama de Colaboración para Inicio del Servidor

De forma general, el comportamiento es el siguiente:

- El usuario indica que se inicie el servidor.
- Se crea el proceso Listener.
- Se recuperan los parámetros de configuración.
- Se recuperan las conexiones JDBC configuradas.
- Se activan las conexiones JDBC.
- Se activa el proceso Listener.

3.3.5 Caso de Uso Consumir Servicios

Esta es una de las principales funciones que debe desempeñar esta aplicación, y se encarga de atender las solicitudes que llegan al servidor de Servicios Web; estas solicitudes recibidas pueden ser de tres tipos:

- Solicitar la descripción de los Servicios Web (documentos WSDL).
- Solicitar la ejecución de una operación para un Servicio Web específico (SOAP).
- Solicitar la página Web de prueba de los Servicios Web configurados (HTML).

Las figuras 3.9 y 3.10 muestran los diagramas de colaboración de acuerdo al escenario que se presente en base a las solicitudes del usuario.

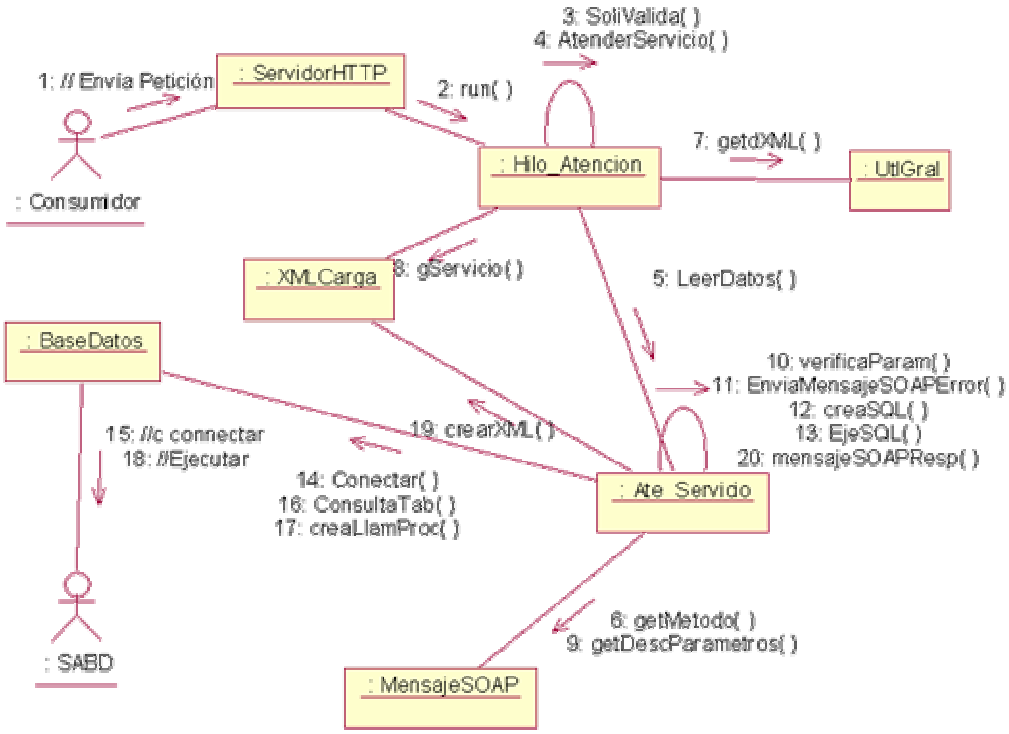


Figura 3.9 Diagrama de Colaboración para Consumir Servicios Web

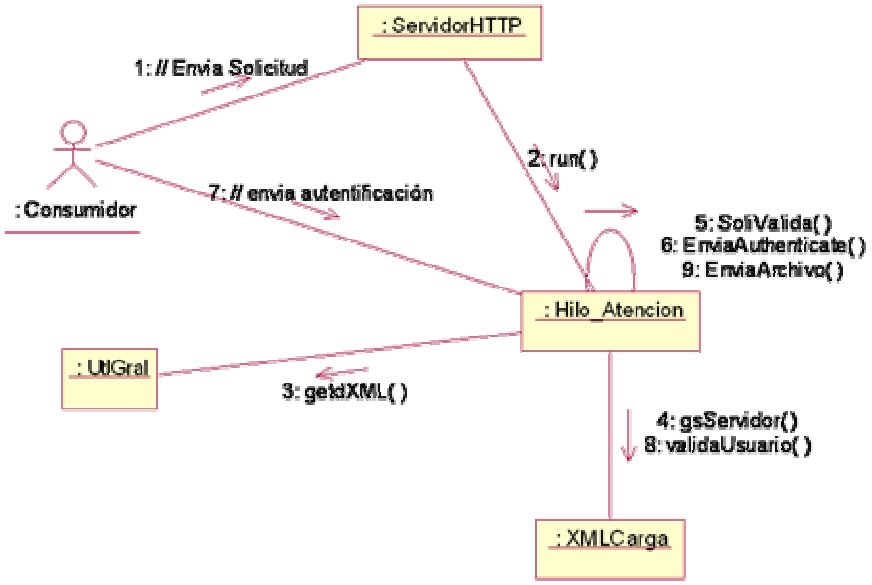


Figura 3.10 Diagrama de Colaboración para Solicitar la Descripción o Página de Prueba

3.3.6 Caso de Uso Configurar Servicios

Al ser una aplicación que brinda las herramientas necesarias para publicar Servicios Web y consumirlos, se debe contar con un módulo a través del cual permita al proveedor de Servicios Web configurarlos con su información general, las operaciones que lo conforman y el detalle de cada operación, además de generar el documento WSDL para cada servicio configurado. El diagrama de colaboración se presenta en la figura 3.11.

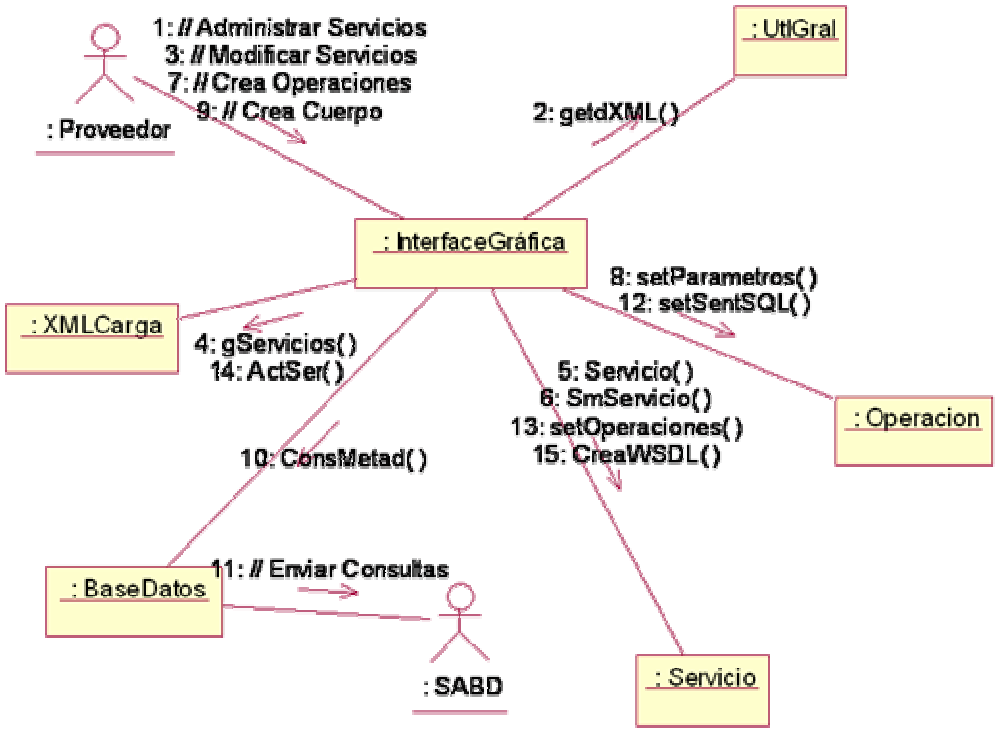


Figura 3.11 Diagrama de Colaboración Configurar Servicios

De manera general, el comportamiento es el siguiente:

- El usuario selecciona Administrar los Servicios.
- Se recupera la información de los Servicios Configurados, en caso de que ya existan.
- Se recupera información sobre las conexiones a los SABD existentes.
- El proveedor modifica la configuración de cada Servicio.
- El proveedor debe modificar o agregar las operaciones para cada Servicio.
- Se conecta a la fuente de datos seleccionada para recuperar los Metadatos.
- El proveedor configura el cuerpo de cada operación.

- Se actualiza la configuración de los servicios.
- Para cada Servicio Web configurado se crea su archivo descriptor.

3.3.7 Caso de Uso Configurar Conexiones SABD

Este módulo permite al proveedor de Servicios Web indicar a la aplicación cuales son los controladores JDBC a utilizar para atender Servicios Web, además de definir los parámetros para cada conexión configurada y su diagrama de colaboración es presentado en la figura 3.12.

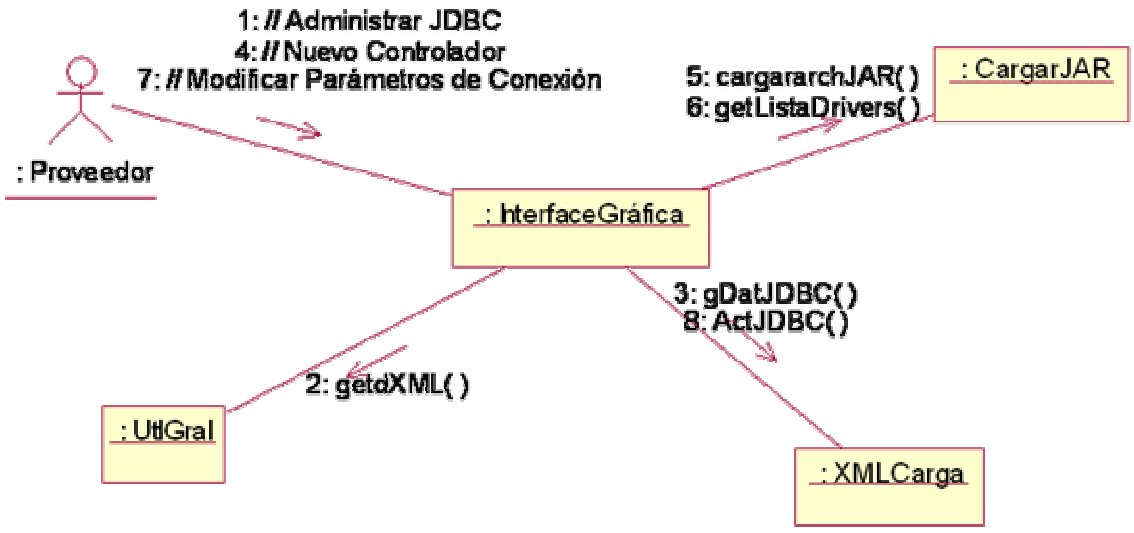


Figura 3.12 Diagrama de Colaboración Configurar Conexiones SABD

De manera general, el comportamiento es el siguiente:

- El usuario selecciona Administrar JDBC.
- Se recupera la información sobre los controladores disponibles y las conexiones ya configuradas.
- Si no existe el controlador requerido, el proveedor puede activarlo indicándole a la aplicación cual es la ubicación de la librería que lo contiene (JAR).
- Se busca los controladores disponibles para esa librería.
- El proveedor puede modificar o configurar nuevas conexiones.
- Se actualiza la configuración XML.

3.4 Modelado de la Aplicación (Diseño)

En esta sección presenta el diseño de los paquetes y clases de los cuales esta compuesta esta aplicación; los diagramas de clase presentados no contienen todas las clases que la componen, ya que muchas de estas clases corresponden a la interfaz gráfica, por lo que sólo se muestran las clases significativas del proyecto. Es importante señalar que en esta parte del documento se presentan las clases significativas, así como los atributos y operaciones relevantes de la aplicación. Para mayor detalle consulte el anexo F.

3.4.1 Paquetes

Las clases que conforman a este proyecto están agrupadas en paquetes. Estos paquetes están construidos en base a lo que representan sus elementos para esta aplicación. El diagrama de paquetes correspondiente se presenta en la figura 3.13. Posteriormente se explican las clases principales por paquete, presentando diagrama de clases, así como una descripción de los principales métodos y atributos.

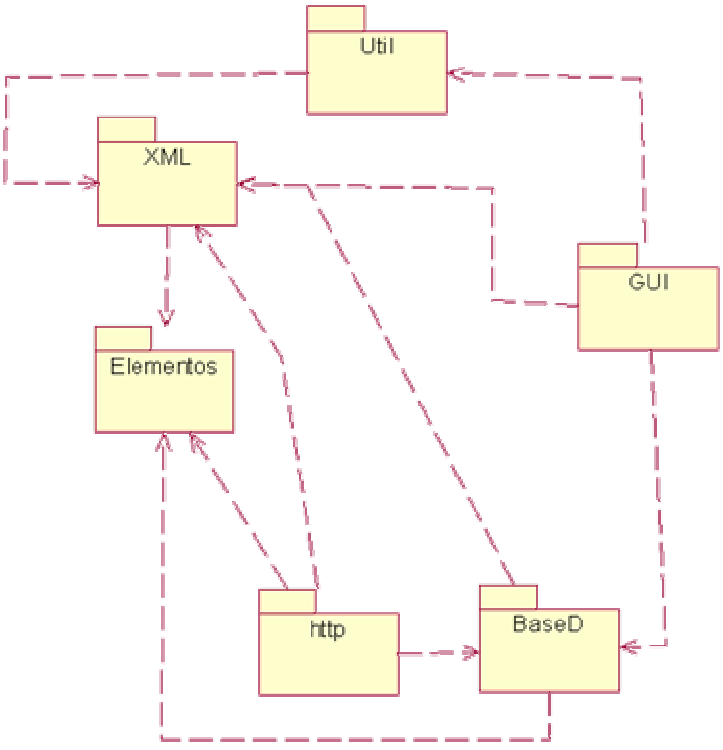


Figura 3.13 Diagrama de Paquetes de la Aplicación

Descripción:

El paquete principal *wsadmin*, contiene la clase principal (*AdmSer*) de esta aplicación.

- **GUI:** este paquete contiene las clases necesarias para el control de la interfaz gráfica
- **Util:** este paquete contiene clases que contienen diferentes métodos, constantes y atributos, los cuales son referenciados por el resto de las clases. Principalmente son clases que sirven de apoyo para el funcionamiento general de la aplicación.
- **Elementos:** este paquete contiene las clases son necesarias para la configuración y operación de la aplicación. Además son utilizadas para poder recuperar la información de configuración de XML.
- **BaseD:** en este paquete existen las clases principales de la aplicación, las cuales están encargadas de la comunicación e interacción con el SABD, además de obtener los metadatos de la base de datos.
- **HTTP:** este paquete contiene las clases necesarias para iniciar el servidor de Servicios Web, además de las encargadas de atender las solicitudes realizadas por los consumidores de Servicios Web.
- **XML:** este paquete contiene las clases para el manejo de XML, se encarga de manipular los archivos de configuración, crear los archivos descriptores de los Servicios Web (WSDL), y el manejo de los mensajes SOAP entre el servidor y los consumidores.

3.4.2 Paquete wsadmin.GUI

Este paquete contiene las clases que se encargan de manejar la administración de la interfaz gráfica, además de las clases que sirven de apoyo para simplificar y facilitar al administrador la tarea de crear Servicios Web, así como clases que le brindan una interacción visual con el SABD.

Este paquete esta agrupado de acuerdo a la funcionalidad de las clases que lo contienen, por lo que existen otros paquetes dentro del mismo.

3.4.3 Paquete wsadmin.XML

Este paquete contiene las clases involucradas en el menaje de documentos basados en XML como son los archivos de configuración de la aplicación para cada Servicio Web, la generación del documento de descripción de cada Servicio Web, las clases necesarias para el manejo de mensajes SOAP y las clases necesarias, para la conversión de los resultados a XML. La figura 3.14 presenta el diagrama de clases.

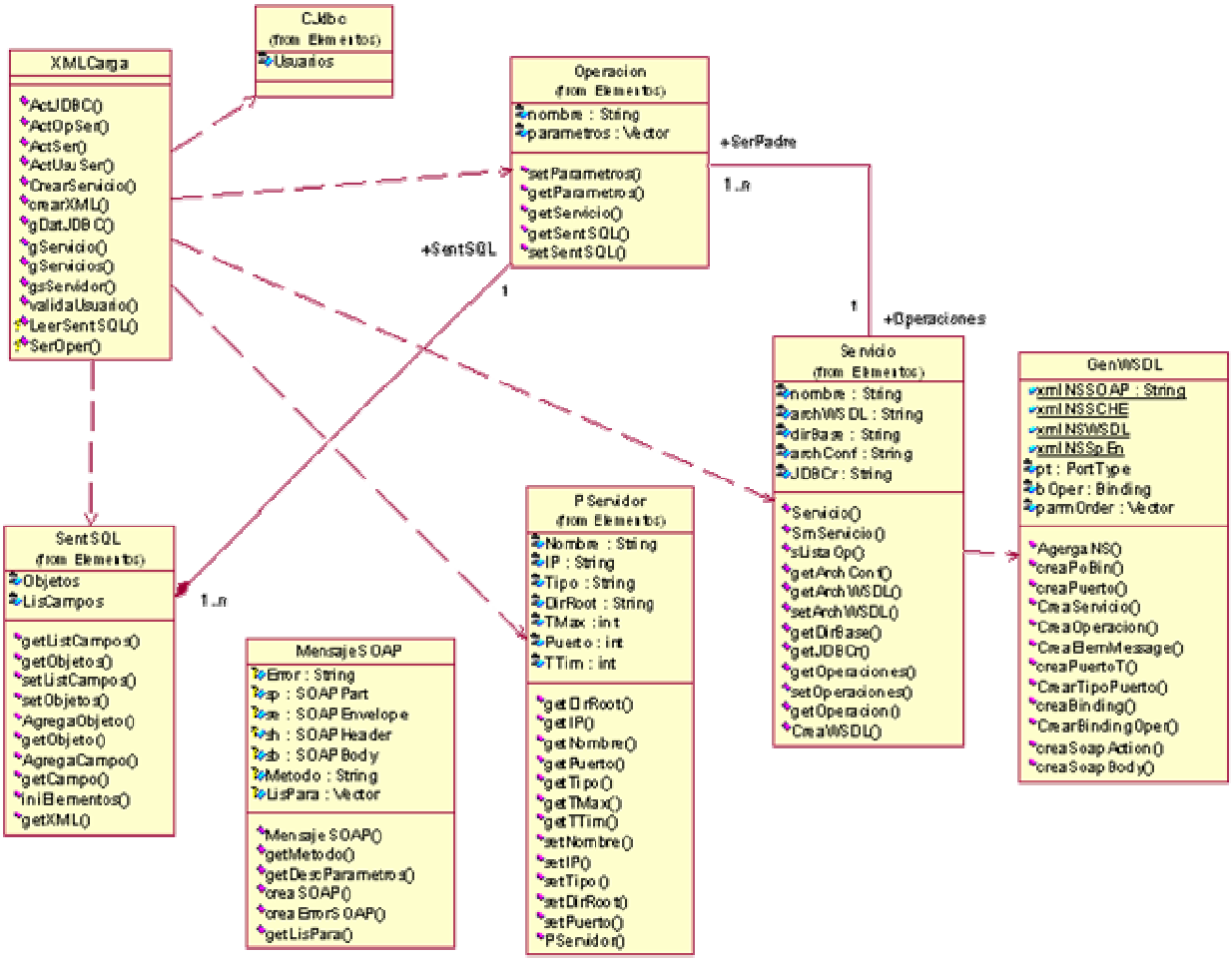


Figura 3.14 Diagrama de Clases para wsadmin.XML

3.4.4 Paquete wsadmin.Util

Este paquete contiene las clases necesarias para constantes y eventos generales de la aplicación, también cuenta con clases para manipular los archivos JAR y que estos puedan ser contemplados por la aplicación en tiempo de ejecución y al inicio de la aplicación. El diagrama de clases es mostrado en la figura 3.15.

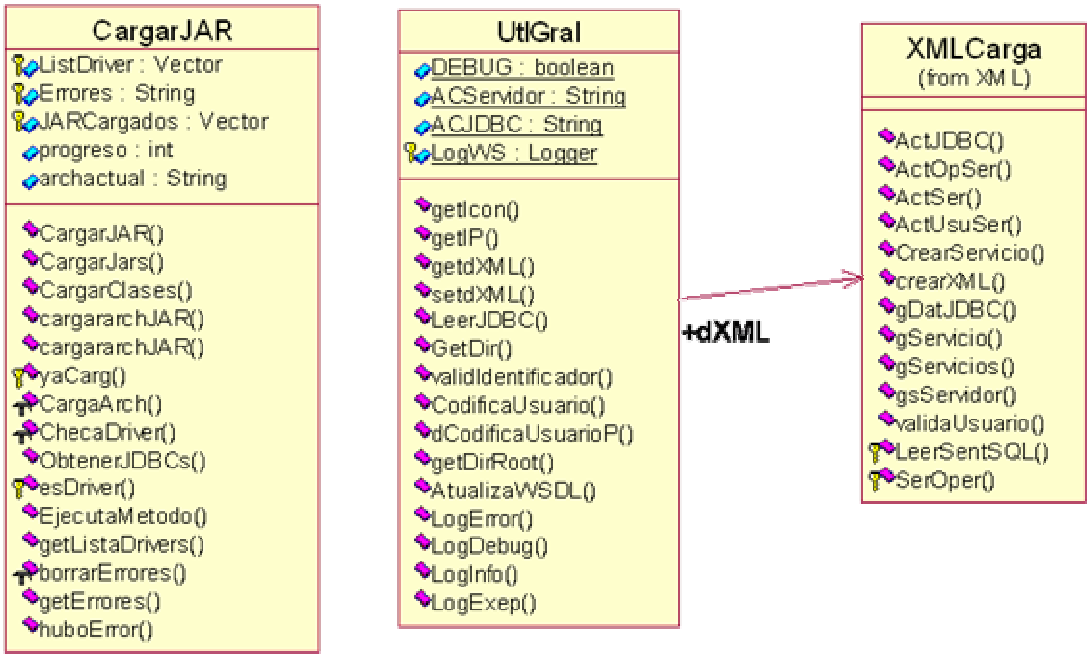


Figura 3.15 Diagrama de Clases para wsadmin.Util

3.4.5 Paquete wsadmin.Elementos

Las clases que componen este paquete corresponden a clases que son utilizadas durante la ejecución de la aplicación y que son comunes a la mayoría de los módulos que la componen Estos elementos en su mayoría corresponden a los elementos que se encuentran en los archivos de configuración XML. El diagrama de clases es mostrado en la figura 3.16.

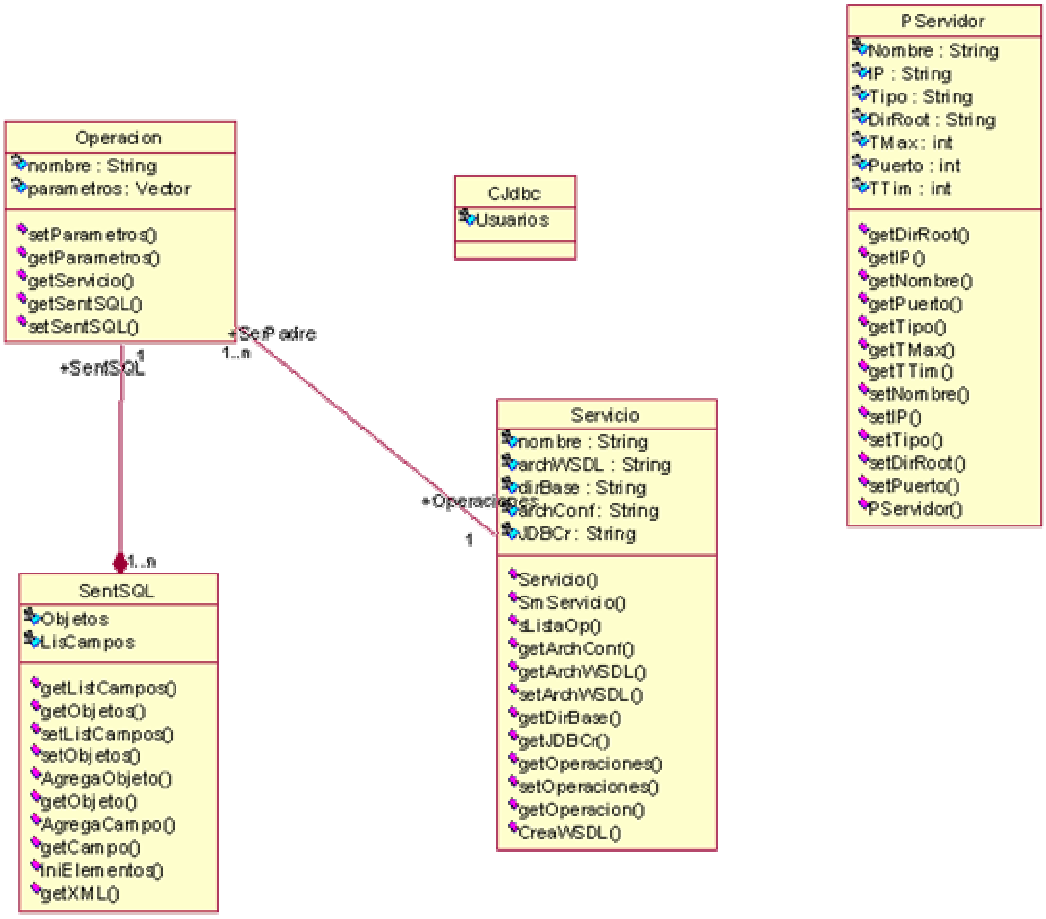


Figura 3.16 Diagrama de Clases para wsadmin.Elementos

3.4.6 Paquete wsadmin.http

Uno de los módulos principales en esta aplicación es el que se encarga de publicar los servicios y atender las solicitudes enviadas por los consumidores de los mismos, y en donde se concentran las clases principales que se encargan del control de este módulo y que están relacionadas con el manejo de Servicios Web. El diagrama de clases es mostrado en la figura 3.17.

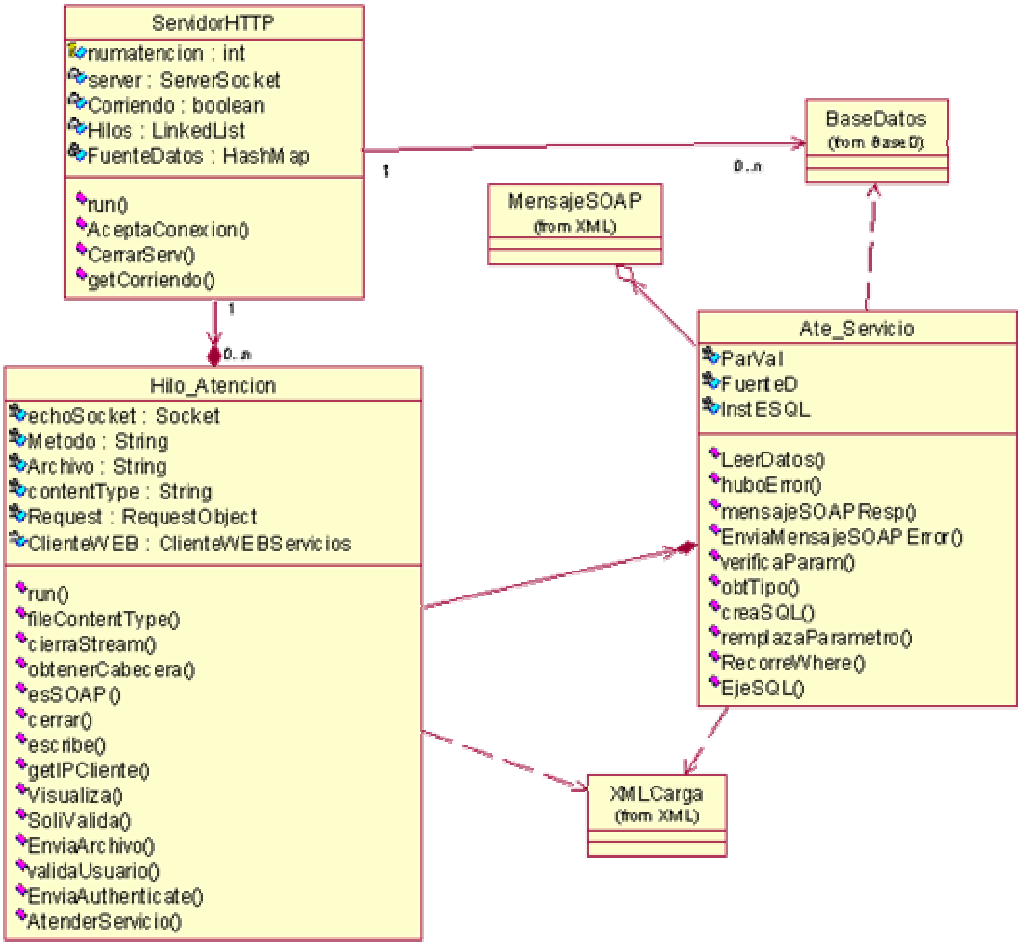


Figura 3.17 Diagrama de Clases para wsadmin.http

3.4.7 Paquete wsadmin.BaseD

Este paquete brinda la clase principal encargada de manejar e interactuar con el SABD, brindando métodos para acceder a la información particular que almacena; esta clase debe conocer los parámetros de conexión, así como las constantes necesarias para esta interacción. El diagrama de clases es mostrado en la figura 3.18.

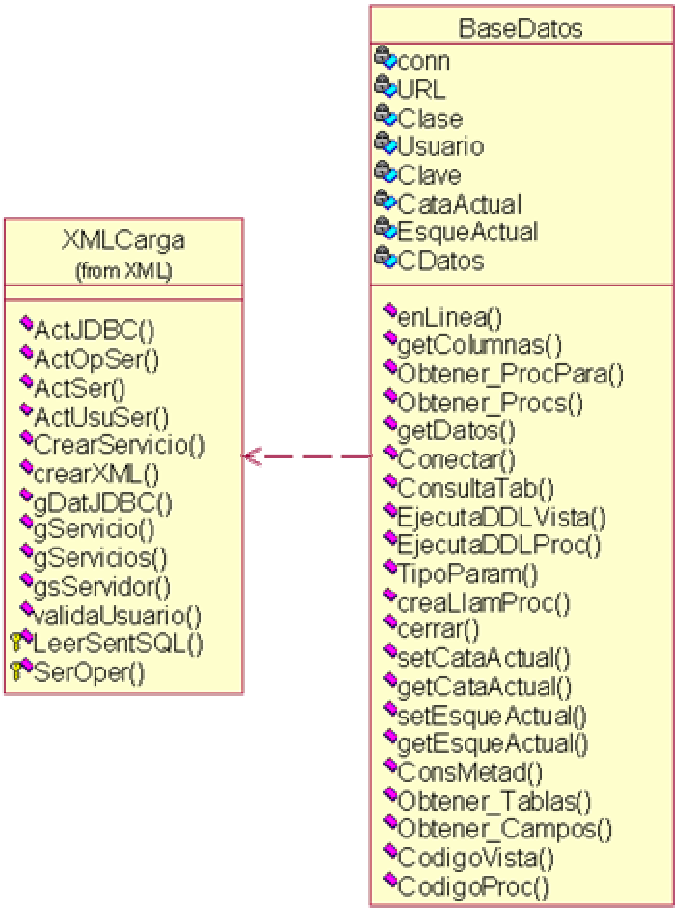


Figura 3.18 Diagrama de Clases para wsadmin.BaseD

3.5 Pantallas Principales de la Aplicación

Como se menciona en capítulos anteriores, uno de los objetivos de esta aplicación es ser simple en su administración, por lo que a continuación se muestran las principales pantallas para poder crear los Servicios Web.

3.5.1 Pantalla Principal

Para poder controlar el flujo de creación de Servicios Web, se cuenta con la pantalla principal de la aplicación, presentada en la figura 3.19.

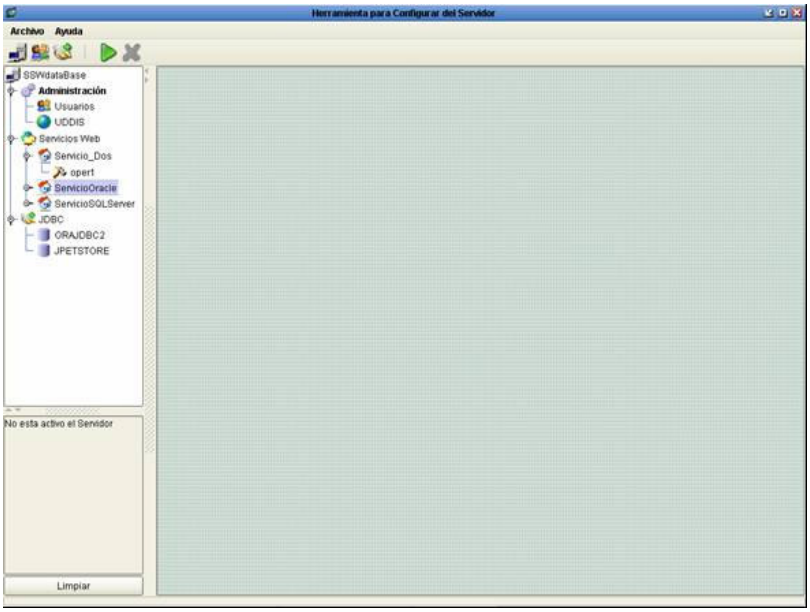


Figura 3.19 Pantalla Principal

Las partes principales de la pantalla son los siguientes:

- **Panel de Contenido:** en la parte izquierda se muestra en forma jerárquica las opciones que son configurables para la aplicación.
- **Panel de Visualización:** en la parte central, en su caso, se muestran las opciones del elemento seleccionado en la estructura de contenido.
- **Barra de botones:** ubicada en la parte superior por debajo del menú de aplicación, contiene algunas opciones para acceder rápidamente a la configuración, además de agregar algunas opciones que no se presentan en la estructura de contenido.
- **Panel de Estado del Servidor:** ubicado en la parte inferior izquierda de la pantalla, muestra los mensajes de envío y recepción del servidor web, siempre y cuando este activo.

3.5.2 Pantalla de Configuración de Servidor Web

Al contar con un servidor de Servicios Web, se debe ofrecer la forma de configurarlo. La figura 3.20 muestra la pantalla donde el administrador podrá determinar los parámetros necesarios para poder activar el servidor.

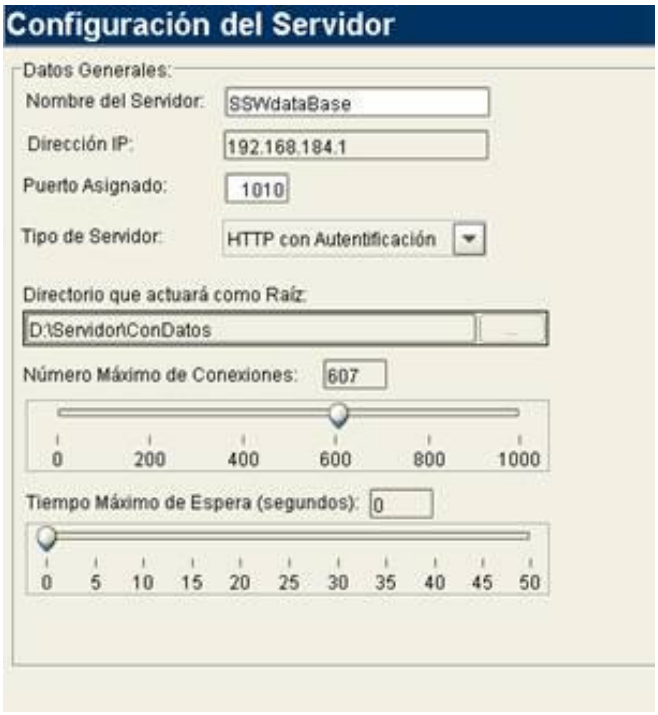


Figura 3.20 Pantalla para Configurar el Servidor de Servicios Web

En esta pantalla el administrador definirá los parámetros necesarios para inicializar a la aplicación, como son el punto de acceso a los Servicios Web (Dirección IP y puerto), así como el comportamiento que debe tener el servidor al momento de atender las solicitudes (si el usuario debe autentificarse o no), entre otros.

3.5.3 Pantallas para la Configuración de Servicios Web

Para crear Servicios Web, se debe definir su configuración general del servicio, el detalle de cada servicio, así como el cuerpo de cada operación. Al cambiar los datos generales de los servicios y la composición de cada operación se genera o actualiza el archivo de descripción para cada Servicio Web (WSDL).

3.5.3.1 Configuración General de Servicio

La pantalla mostrada en la figura 3.21 es utilizada para que el administrador pueda asignar las características de cada servicio, entre las que destacan el nombre y la conexión JDBC a utilizar para poder dar atención a las solicitudes enviadas por los consumidores.

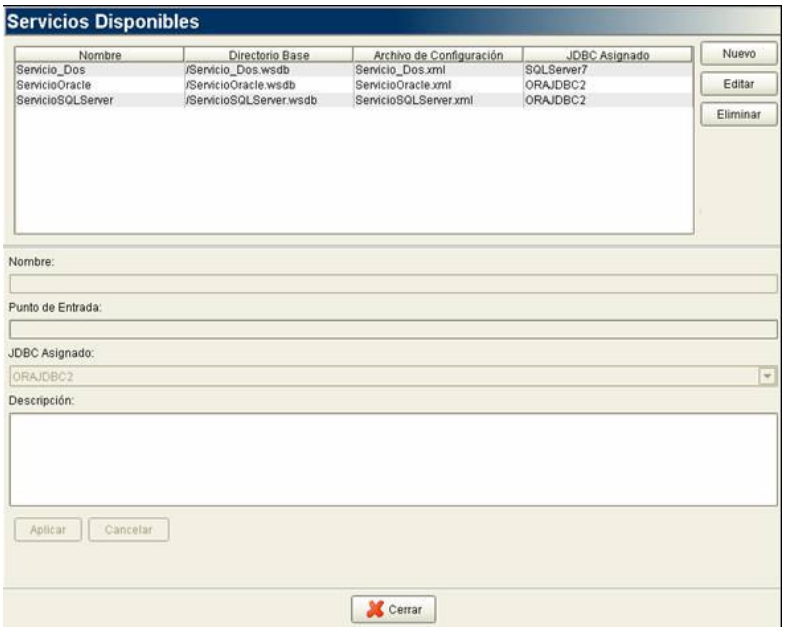


Figura 3.21 Pantalla para Administrar Servicios Web

Esta pantalla se debe mostrar al momento de seleccionar el elemento servicios en la estructura de contenido de la pantalla principal.

3.5.3.2 Configuración del Detalle del Servicio

En la estructura de contenido se muestran los servicios que existen actualmente, y en donde al seleccionar alguno de ellos se muestra la pantalla de la figura 3.22, ayudando al administrador a definir el detalle de cada servicio.

En esta pantalla el administrador define las operaciones de las cuales estará compuesto el servicio, además de definir los parámetros por cada operación.

Figura 3.22 Pantalla para Administrar Operaciones por Servicio Web

3.5.3.3 Configuración del Cuerpo de la Operación

La pantalla de la figura 3.23 esta diseñada para simplificar la creación del cuerpo de cada una de las operaciones que contienen los Servicios Web. En esta pantalla se construyen las sentencias SQL que la operación solicitará ejecutar al SABD. Como se muestra en la figura 3.23, las sentencias SQL están clasificadas de acuerdo el tipo de instrucción, además se muestran los elementos contenidos en la fuente de datos asignada al Servicio Web de acuerdo a la pestaña correspondiente (vistas, tablas, procedimientos y/o funciones).

El usuario no introduce sentencias SQL en forma directa, ya que sólo selecciona el tipo de instrucción y los elementos que estarán involucrados, además de establecer si esta instrucción informará los resultados obtenidos al consumidor en XML.

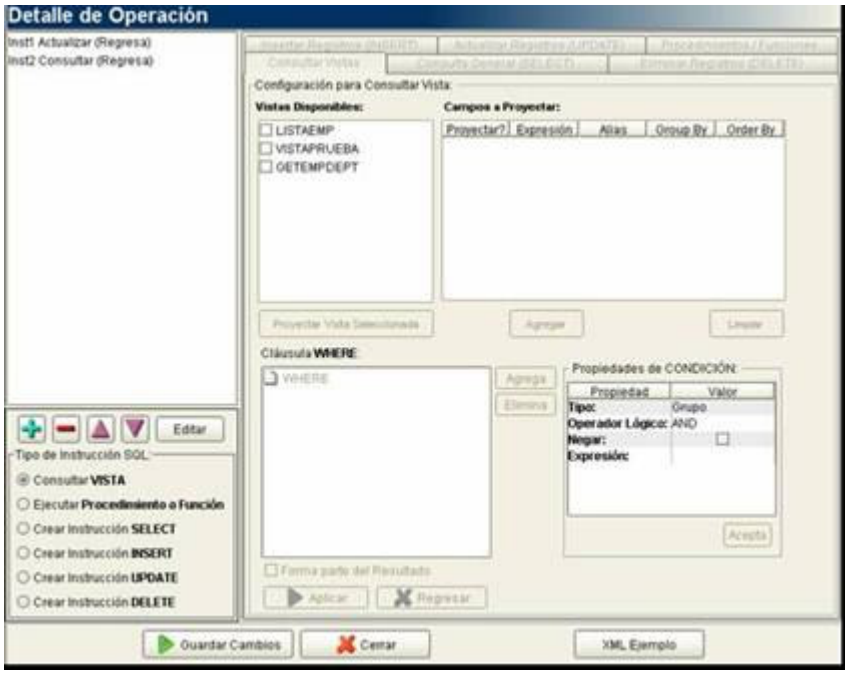


Figura 3.23 Pantalla para Asignar Instrucciones SQL a Operaciones

Cada operación de los Servicios Web configurados puede contener una o más sentencias SQL; cada una de ellas es enviada al SABD y en base a los resultados arrojados y de acuerdo a la configuración de la instrucción, se crea el documento XML que será enviado al consumidor de los Servicios Web.

3.6 Archivos de Configuración

De acuerdo a los requerimientos establecidos en el capítulo anterior, la configuración de la aplicación se ha distribuido en tres archivos principales:

3.6.1 Configuración de Servidor

La configuración del servidor de Servicios Web está contenida en el archivo principal de configuración (conf.xml), en el se encuentran los datos del servidor, las características principales de los Servicios Web, así como las características de cada conexión a los SABD; además podemos encontrar a los usuarios que pueden hacer uso de los servicios y al usuario administrador que puede modificar estas configuraciones.

3.6.1.1 Configuración Servidor (ConfSer)

Los siguientes son los campos necesarios para poder configurar el servidor; de ellos depende que se generen de manera correcta los archivos descriptores de cada Servicio Web.

Campo	Descripción
nombre	Indica el nombre descriptivo del servidor e indica que es un campo informativo para la aplicación y que debe estar presente.
dirIP	Indica la dirección IP del equipo que cuenta la aplicación y que inicialmente contiene el valor "INDEFINIDA"; cuando se configura el servidor se obtiene la dirección IP y es actualizado el campo, además esta dirección será la que servirá como punto de acceso para la atención de solicitudes, por lo que debe estar presente.
puerto	Este campo le indica a la aplicación el puerto en el que estará escuchando peticiones el servidor de Servicios Web, este campo debe estar presente.
tipo	Este campo determina la forma en que pueden ser consumidos los Servicios Web. Los tipos válidos para este campo son HTTP y HTTP con autentificación, en donde este ultimo solicitará un usuario y clave de acceso al consumidor de los Servicios Web.
directorio	Indica al servidor donde debe buscar los archivos de configuración, así como los archivos de descripción de los servicios.
MaxCon	Este campo indica el máximo número de conexiones que podrá soportar el servidor. El máximo permitido es 1000, sin embargo dependerá de la configuración del equipo donde esta aplicación sea instalada.
TiempoSesion	Este campo indica el tiempo en que se estará verificando el puerto donde escucha el servidor de Servicios Web para atender las solicitudes.

3.6.1.2 Configuración de Usuarios

Este elemento debe contener los usuarios que estarán disponibles para acceder a la aplicación y los usuarios que podrán hacer uso de los Servicios Web.

Campo	Descripción
nombre	Indica el nombre del usuario.
usrID	Indica el usuario con el que será identificado el nombre anterior y es el que deberá proporcionar el consumidor de los Servicios Web cuando el servidor este configurado con autentificación.
clave	Determina la clave de acceso al sistema, que esta codificada usando el tipo de codificación base64, es utilizada por el método de autentificación básica definido en el protocolo HTTP.
tipo	Determina el tipo de usuario; los valores permitidos son ADM como administrador y VIE como usuario de Servicios Web.

3.6.1.3 Configuración de Servicios (servicios)

En este elemento se muestra los datos generales de la configuración de cada Servicio Web, y que consta de los siguientes elementos:

Campo	Descripción
nombre	Indica el nombre del servicio; este campo se utilizará para generar el archivo de descripción y el archivo de configuración.
dirBase	Indica el punto de acceso al Servicio Web, utilizado en el momento de generar el archivo de descripción y que identifica al servicio se está solicitando ejecutar.
archConf	Determina en donde se debe buscar el detalle del Servicio Web; este campo es generado automáticamente por la aplicación en base al atributo nombre.
Descripcion	Este elemento contiene la descripción del Servicio Web, sin límite de caracteres.

3.6.1.4 Configuración de Conexiones (conexiones)

En este elemento se presenta la configuración de cada conexión que será utilizada para acceder a los SABD a través de JDBC.

Campo	Descripción
nombre	Indica el nombre de la conexión JDBC, que será referenciado cuando se asigne a un Servicio Web.
url	Este dato dependerá de cada controlador JDBC, y en el se muestra la información de conexión al SABD.
clase	Indica a la aplicación cual es la clase java del controlador JDBC que será utilizada para poder activar esta conexión.
SerVis	Indica la sentencia SQL que deberá ser utilizada para poder recuperar el código SQL de una vista.
SenPro	Indica la sentencia SQL que deberá ser utilizada para poder recuperar el código SQL de un procedimiento o función de la base de datos.

A su vez este elemento cuenta con un subelemento que indica el usuario que será utilizado para conectarse al SABD, en donde sus atributos son:

Campo	Descripción
usrJDBC	Id del usuario para conectarse a la base de datos.
clave	Indica la clave codificada en Base64 del usuario.

3.6.2 Configuración de apoyo para JDBC

El archivo de configuración *confjdbc.xml* contiene la información sobre los controladores que están disponibles para el funcionamiento de la aplicación, así como algunas características que comúnmente se utilizan.

URL

En este elemento se presentan dos atributos que sirven para simplificar la configuración de las conexiones JDBC.

Campo	Descripción
clase	Indica la clase java que estará ligada al URL que a continuación se presente.
url	Indica un patrón de URL que puede ser utilizado para la clase que se seleccionó y al cual se le establecen los parámetros correspondientes; este elemento se usa como ayuda para la configuración de las conexiones JDBC en la aplicación y dependerá del SABD y del controlador JDBC elegidos.

Driver

Presenta los archivos JAR de los controladores JDBC que están disponibles y que deberá cargar la aplicación para poder configurar conexiones JDBC y Servicios Web.

Campo	Descripción
path	Indica de forma absoluta la ubicación del archivo JAR que contiene el controlador JDBC.
estado	Indica si el controlador JDBC está o no disponible; este campo se actualiza al momento de iniciar la carga de los controladores JDBC.

Patron

La existencia de este elemento principalmente va orientada a auxiliar al usuario de la aplicación al momento de construir funciones o procedimientos almacenados, mostrando una plantilla de cómo debe ser generado.

Campo	Descripción
db	Indica el SABD elegido sobre el cual se presentan las plantillas de función y procedimientos almacenados.

Para definir las plantillas de las funciones y/o procedimientos, se definen dos subelementos:

Campo	Descripción
Funcion	Contiene la plantilla para poder construir funciones.
Procedimiento	Contiene la plantilla para poder construir procedimientos almacenados.

La existencia de estos elementos dependerá de las características del SABD. Por otro lado la existencia de los elementos usados como patrones deberá ser en forma manual, pero en la aplicación se incluyen las plantillas de los SABD que se utilizarán de prueba (SQL Server, Oracle y PostgreSQL).

3.6.3 Configuración del Detalle de los Servicios

En este archivo de configuración (*nombreServicio.xml*, en donde *nombreServicio* corresponde al identificador del Servicio Web) se define el detalle para el servicio al cual pertenece este archivo, los archivos de configuración por servicio se crean al momento de utilizar por primera vez el servidor de Servicios Web y se denominan con el nombre elegido para el servicio y con la extensión XML (OracleSW.xml).

3.6.3.1 Elemento Servicio

Es el elemento principal y contiene los elementos y atributos necesarios para poder configurar un servicio y poder dar atención a las solicitudes que realicen los consumidores.

Campo	Descripción
archWSDL	Indica el archivo descriptor del servicio al que pertenece el archivo.
JDBC	Indica que conexión JDBC utilizará el servicio para enviar las instrucciones SQL de las cuales se componen las operaciones.

3.6.3.2 Elemento Operacion

Cada Servicio Web podrá contener una o más operaciones; en esta parte se muestra el detalle de las operaciones.

Campo	Descripción
nombre	Indica como será conocida la operación en el archivo de descripción.

El cuerpo de cada operación esta formado con instrucciones SQL, por lo que para cada operación existirá un conjunto de elementos que definen a cada operación SQL:

Elemento opeSQL

Este elemento nos describe el detalle de la operación SQL:

Campo	Descripción
nombre	Sirve para identificar la sentencia dentro del cuerpo de cada operación.
tipo	Indica el tipo de sentencia SQL que será ejecutada.
retresult	Indica cuando se devolverá el resultado en XML al consumidor.

La configuración de la sentencia SQL dependerá del tipo de sentencia que se este construyendo; los elementos principales son los siguientes:

Campo	Descripción
parametro	Define cada uno de los parámetros de la operación que serán enviados por el consumidor del Servicio Web.
ObjBD	Muestra una lista de los objetos o elementos de la base de datos que estarán involucrados para construir la instrucción SQL.
Lista_Campos	Muestra un conjunto de campos involucrados en el objeto o elemento de la base datos, aquí se determina que campos son devueltos al ejecutar la instrucción SQL.
Where	Muestra una lista de condiciones para la sentencia a ejecutar, además de agrupar condiciones y poder definir el tipo de operador lógico a utilizar.
Condicion	Representa el detalle de cada condición que estará involucrada en las sentencias *where* de las sentencias SQL que así lo requieran.

3.7 Resumen

En este capítulo se presentó el análisis de la Interfaz de Publicación y del Servidor HTTP construida en este proyecto. Se mostraron los diagramas de caso de uso y los diagramas de colaboración. Además se presentó la estructura de la aplicación, algunas de las pantallas principales de la misma y la interacción entre las clases que la conforman, presentando la composición de los archivos de configuración.

En el capítulo siguiente se muestran con más detalle el comportamiento de las clases presentadas en este capítulo, además se da una descripción de los principales métodos que las componen.

Capítulo IV
Implementación

Actualmente la reutilización de componentes y/o código ha tenido un auge importante en el desarrollo de aplicaciones (de cualquier tipo), por ello es necesario conocer la implementación de dichas aplicaciones, así como la descripción de las API (Application Programing Interface, Interfaz para programas de aplicación) utilizadas y el código desarrollado de los módulos principales de la aplicación.

4.1 Codificación de la Interfaz

Para cumplir los requerimientos establecidos en capítulos anteriores se eligió el lenguaje de programación Java como plataforma de desarrollo y el cuerpo de los Servicios Web estará formado por sentencias SQL que serán enviadas al SABD asignado a cada Servicio Web.

Para llevar a cabo la implementación de este proyecto se agruparon las diversas clases que lo conforman de acuerdo al funcionamiento de cada una de ellas; este agrupamiento forma el árbol de directorios que se presenta en la figura 4.1.

Directorio	Descripción
wsdamin	Paquete principal del desarrollo de la aplicación, contienen la clase principal de Java que se encarga de ejecutar la aplicación.
BaseD	Contiene la clase de Java que se encarga de interactuar con el SABD.
Elementos	Contiene clases que son utilizadas para recuperar y almacenar información en las archivos de configuración y con el SABD.
GUI	Agrupa otros paquetes que se encargan de la interfaz gráfica mostrada al usuario.
AmbGraf	Contiene las clases de la pantalla principal de la aplicación y la clase para seleccionar los controladores JDBC.
ConfServidorWeb	Archivos Java para proporcionar al usuario la interfaz necesaria para configurar el servidor de Servicios Web.
InstSQL	Contiene las clases que controlan la creación del cuerpo de las operaciones para cada Servicio Web.
JDBC	Clases Java para definir las conexiones a utilizar y clases que permiten interactuar con el SABD seleccionado.
ObjGraf	Contiene paquetes que se utilizan como componentes de otras clases.
Etiquetas	Componente en Java para la presentación de etiquetas con un fondo degradado.
Formatos	Clases Java que contienen los formatos de visualización de las tablas utilizadas en esta aplicación.
ListCheck	Componente Java que permite agregar funcionalidad a la Clase JListBox.
OProp	Componente Java que se utiliza como parte de las tablas de propiedades en InstSQL y que permite visualizar un diálogo para obtener información del usuario.
ServiciosWeb	Clases Java para la administración de los Servicios Web, tanto las características generales como administrar cada una de las operaciones de cada Servicio Web.
http	Clases Java que se encargan del control del servidor de Servicios Web.
Iconos	Contiene los archivos gráficos utilizados en la interfaz gráfica.
Util	Diversas clases que contienen métodos utilizados constantemente por el resto de las clases.

XML	Contiene clases Java que se encargan de interactuar con los archivos de configuración, así como el manejo de mensajes SOAP y la generación de los archivos WSDL de cada Servicio Web.

Figura 4.1 Árbol de Directorio (paquetes) del Proyecto de JBuilder para el Desarrollo de la Aplicación

El paquete wsadmin contiene a la clase *AdmSer* la cual es la clase principal de la aplicación y al ser invocada debe realizar las siguientes acciones:

- Crea un archivo para enviar mensajes al usuario en caso de asignar a la variable DEBUG el valor TRUE, la cual se encuentra en la clase *wsadmin.Util.UtlGral*, que por defecto tiene el valor FALSE y puede ser cambiado al momento de ejecutar la aplicación.
- Verificar la versión de la máquina virtual de Java, la cual deberá ser la 1.4 o superior; en caso de que sea una versión anterior, se envía un mensaje indicando que no cuenta con la versión adecuada para poder utilizar esta aplicación.
- Cargar los controladores JDBC y librerías necesarias (archivos JAR) para la utilización de la aplicación; en caso de que exista algún error en la carga de estas librerías se le indicará al usuario y no podrá continuar la ejecución de la aplicación.
- Leer el archivo de configuración general de la aplicación y en base a su contenido, leer la configuración de cada Servicio Web construido; en caso de existir algún error al momento de cargar estos archivos, se indicará al usuario con un mensaje en pantalla y no podrá continuar la ejecución de la aplicación.
- Crear los objetos necesarios para el manejo de la interfaz gráfica (*wsadmin.GUI.AmbGraf.FPrin*).

En las secciones siguientes se presenta la implementación de los módulos principales para el funcionamiento de la aplicación. Para ver el detalle de cada una de las clases aquí descritas consulte el anexo E que muestra el código fuente completo de la aplicación.

4.1.1 Codificación del Módulo XML

Al elegir a XML como formato para la configuración y para el intercambio de información, se desarrolló un módulo que se encarga de interactuar con los archivos de configuración, y que por otro lado se encarga de recibir los datos arrojados por las sentencias SQL enviadas al SABD y convertirlas en XML para posteriormente comunicarlo al consumidor del Servicio Web.

4.1.1.1 Wsadmin.XML.XMLCarga

Esta clase es la que implementa el módulo de XML y contiene los métodos necesarios para poder manipular los archivos de configuración y permitir al usuario de la aplicación administrar todos los elementos necesarios para la creación de Servicios Web, además de brindar la posibilidad de recuperar la información de cada elemento. Es por ello que existen diversas clases que le permiten a este módulo, a través de sus métodos, comunicar los resultados de la información leída de los archivos de configuración.

El control y acceso a XML se realiza a través de la API DOM (Document Object Model, Modelo de Objetos de Documentos) permitiendo desplazarse a través de todo el archivo XML. Adicionalmente se utiliza XPATH (sección 4.3) para poder realizar búsquedas de los elementos de acuerdo a un atributo que identifique al elemento deseado.

Los métodos que se encargan de recuperar el detalle de cada elemento existente en los archivos de configuración dependerán del tipo de elemento; si existe sólo un elemento de este tipo, no será necesario tener un parámetro para poder identificar dicho elemento.

Código General

```
…
  NodeIterator nl =   Buscar_Nodos_que_cumplan_con("cadena XPATH", Dom );
  Si existen nodos entonces
  {
    …
    // Recupera el detalle en base a sus atributos o valores
    …
  }
  Si los objetos recuperados son válidos entonces
  {
    …
    crear objeto según elemento buscado -> conj
    return(conj);
  }
}
Si existen errores
{
  Indicar el error ocurrido
}
regresar
```

El código de ejemplo anterior muestra la forma en que es codificado cada uno de los métodos que se encargan de recuperar información de algún archivo XML.

A continuación se presenta una lista de métodos que son ocupados para la interacción con los archivos XML.

Método	Descripción
public void ActJDBC(String nomant, String nomact, String url, String clase, String usuar, String clave, String senvis, String senpro, int nue)	Este método se encarga de actualizar o agregar una conexión JDBC en base a los parámetros recibidos.
public boolean ActOpSer(Operacion op)	Realiza una actualización de una operación específica en un Servicio Web.
public void ActSer(PServidor ser)	Actualiza la configuración del servidor de Servicios Web.
public void ActualizaJar(String jar, String tipo)	Modifica el estado actual de un controlador JDBC.
public void ActualizaListJar(String path)	Modifica la lista de controladores JDBC disponibles.
public void ActUsuSer(javax.swing.JTable TabUsu)	Modifica y actualiza la lista de usuarios de la aplicación.
public void CrearServicio(String nombre, String dir, String jdbc, String descrip, int tipoop)	Crea un archivo de configuración XML para un Servicio específico.
public Cjdbc gDatJDBC(String JDBC)	Obtiene las características de una conexión JDBC.
public Vector getJARS(String tipo)	Obtiene los controladores JDBC disponibles.
public Servicio gServicio(String nombre, boolean leeop)	Obtiene las características generales y/o el detalle de cada Servicio Web identificado por el nombre.
public Servicio gServicios(int num, boolean leeop)	Obtiene las características generales y/o el detalle de cada Servicio Web identificado por el número de Servicio.
public PServidor gsServidor()	Obtiene las características del servidor de Servicios Web.
public UsuAdm gUsuAdm(int numusu)	Obtiene los datos del usuario en base al número de usuario.
public UsuAdm gUsuAdm(String nomusu)	Obtiene los datos del usuario en base al nombre de usuario.
public boolean validaUsuario(String usid, String uspass)	En base a la información recibida por el consumidor, se valida al usuario y su clave para saber si cuenta con permisos.
protected NodeIterator BuscaNodos(String Path, Document domi)	Se encarga de realizar las búsquedas en XML a través de XPATH.
protected Vector LeerObjSQL(Document domi, String nomOp, String nomSen)	Se encarga de leer los datos de los elementos que deberán ser incluidos en una sentencia SQL.

Es de esta manera que las clases que necesitan información de XML pueden presentar al usuario el contenido de uno o varios elementos en el archivo de configuración XML, además de almacenar los cambios realizados por el usuario.

4.1.2 WSDL

Al ser una interfaz para la creación de Servicios Web, debe generar la descripción de cada uno de los Servicios Web que aquí se construyan para que los consumidores puedan conocer como deberán ser utilizados, los tipos de datos a enviar y que regresarán como resultado de su ejecución.

4.1.2.1 Wsadmin.XML.GenWSDL

Esta clase se encarga de generar la descripción de cada Servicio. Si los cambios realizados al Servicio corresponden al cuerpo y no a la definición del Servicio no será necesario generar el WSDL del Servicio, ya que no se cambia lo que el usuario necesita conocer para consumirlo.

Por otro lado se debe cumplir con la especificación del formato WSDL definida en [L-11] y que se presenta en forma general en el anexo B.

La figura 4.2 presenta la configuración de un Servicio Web generada por el uso de ésta aplicación; en la figura 4.3 se muestra el archivo WSDL generado en base a su configuración y el cual es generado en forma automática.

Archivo de Configuración

```
<?xml version="1.0" encoding="UTF-8"?>
<servicio nombre="ServicioOracle" archWSDL="ServicioOracle.wsdl" publicar="F" publicado="F"
archEjemplo="ServicioOracle.html" JDBC="ORAJDBC2">
    <operaciones>
        <operacion nombre="GetEmpleados">
        </operacion>
        <operacion nombre="GetDeptos">
        </operacion>
        <operacion nombre="getDatporEmp">
            <parametro nombre="idEmp" tipo="int" longd="4" longe="0"/>
        </operacion>
        <operacion nombre="GetCatDep">
        </operacion>
        <operacion nombre="InsertaEmp">
            <parametro nombre="nombre" tipo="string" longd="10" longe="0"/>
            <parametro nombre="FechCont" tipo="date" longd="1" longe="0"/>
            <parametro nombre="Salario" tipo="float" longd="7" longe="2"/>
            <parametro nombre="idDep" tipo="int" longd="4" longe="0"/>
            <parametro nombre="Puesto" tipo="string" longd="10" longe="0"/>
        </operacion>
        <operacion nombre="GetDatporDep">
            <parametro nombre="idDep" tipo="int" longd="4" longe="0"/>
        </operacion>
        <operacion nombre="InsertaDepto">
            <parametro nombre="Nombre" tipo="string" longd="10" longe="0"/>
            <parametro nombre="Actividades" tipo="string" longd="200" longe="0"/>
            <parametro nombre="idUbic" tipo="int" longd="4" longe="0"/>
            <parametro nombre="idProy" tipo="int" longd="4" longe="0"/>
        </operacion>
    </operaciones>
</servicio>
```

Figura 4.2 Archivo de Configuración de un Servicio Web.

En la figura 4.2 podemos encontrar al Servicio Web denominado “ServicioOracle” identificado de forma única dentro de la aplicación, y que contiene siete operaciones, las cuales son:

- **GetEmpleados:** que se encarga de obtener una lista de empleados
- **GetDeptos:** brinda una lista de departamentos
- **getDatporEmp:** esta operación debe buscar los datos específicos de un empleado, a través del parámetro *idEmp* indicando cual empleado se están buscando sus datos.
- **GetCatDep:** Obtiene los datos de los departamentos en la base de datos con una configuración diferente a la que se presenta en la operación *GetDeptos*.
- **InsertaEmp:** esta operación toma los parámetros que envía el consumidor y los ajusta en las sentencias SQL para poder insertar un nuevo empleado en la base de datos.
- **GetDatporDep:** obtiene los datos específicos de un departamento en base al parámetro *idDep*.
- **InsertaDepto:** en base a los parámetros construye las sentencias SQL necesarias para poder dar de alta un departamento en la base de datos.

Con lo anterior se genera el archivo WSDL presentado en la figura 4.3, además de estar disponible a través de una dirección URL para brindarle al consumidor la facilidad de obtener la descripción de cada uno de los Servicios Web.

Archivo WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ServicioOracle="http://new.webservice.namespace"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://new.webservice.namespace">
    <wsdl:message name="DeptxUbiInput">
 </wsdl:message>
    <wsdl:message name="getDatporEmpInput">
        <wsdl:part name="idEmp" type="xsd:int"/>
    </wsdl:message>
    <wsdl:message name="GetDatporDepOutput">
        <wsdl:part name="outXML" type="xsd:string"/>
…
    </wsdl:message>
    <wsdl:portType name="ServicioOracle_port">
        <wsdl:operation name="GetEmpleados">
            <wsdl:input name="GetEmpleadosInput" message="ServicioOracle:GetEmpleadosInput"/>
            <wsdl:output name="GetEmpleadosOutput" message="ServicioOracle:GetEmpleadosOutput"/>
        </wsdl:operation>
…
    </wsdl:portType>
    <wsdl:binding name="ServicioOracle_binding" type="ServicioOracle:ServicioOracle_port">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetEmpleados">
            <soap:operation soapAction="http://192.168.184.1:1010/WSDataBase/ServicioOracle?GetEmpleados"/>
…
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ServicioOracle_service">
        <wsdl:port name="ServicioOracle_Port" binding="ServicioOracle:ServicioOracle_binding">
            <soap:address location="http://192.168.184.1:1010/WSDataBase/ServicioOracle"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Figura 4.3 Archivo WSDL Generado por la Aplicación.

Este archivo puede ser utilizado por cualquier consumidor de Servicio Web para generar en forma automática los mensajes SOAP necesarios. Los métodos utilizados que componen a esta clase se muestran en la figura 4.4.

Método	Descripción
public GenWSDL()	Crea un archivo WSDL nuevo incluyendo algunos encabezados.
Port creaPuerto(String direccion)	Crear el punto de acceso para cada operación del Servicio Web.
public void CreaOperacion(String NomOper, Vector Params)	Crea los mensajes de entrada y salida para la operación que pasa como parámetro.
void CrearTipoPuerto(PortType pt, Operation oper, Message operEnt, Message operSal, String NomOper)	Crea el elemento *portType* del archivo WSDL.
void CrearBindingOper(Binding bOper, Operation oper, PortType pt, String NomOper)	Crea el elemento *Binding* para el Servicio Web al que se hace referencia.
SOAPOperationImpl creaSoapAction(String NomOper)	Crea el elemento *soapAction* que deberá ser enviado junto con el mensaje SOAP.
SOAPBodyImpl creaSoapBody()	Define la forma que deberá ser creado el cuerpo del mensaje SOAP.

Figura 4.4 Lista de Métodos para la Clase GenWSDL.

4.1.3 SOAP

Otro módulo esencial del proyecto es la utilización de mensajes SOAP para poder interactuar con los consumidores, por lo tanto debe ser capaz de interpretar y validar los mensajes recibidos por los consumidores y debe construir los mensajes SOAP que se enviarán al cliente de los Servicios que podrán ser el resultado de la ejecución de los Servicios Web o de la comunicación de que ocurrió un error durante la ejecución de dicho Servicio.

4.1.3.1 wsadmin.XML.MensajeSOAP

Esta clase se encarga de interpretar los mensajes SOAP recibidos por el consumidor y de generar los mensajes SOAP de respuesta, en la figura 4.5 se muestra un ejemplo de mensaje SOAP recibido.

Mensaje SOAP

```
POST /WSDataBase/ServicioOracle HTTP/1.1Content-Type: text/xml
User-Agent: XML Spy
SOAPAction: "http://192.168.10.1:1010/WSDataBase/ServicioOracle?GetCatDep "
Host: 192.168.10.1:1010
Content-Length: 408
Connection: Keep-Alive
Cache-Control: no-cache
Authorization: Basic bHVpc2dlOnBhc28=

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <m:GetCatDep xmlns:m="urn:WSSABD:ServicioOracle" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 4.5 Ejemplo de Mensaje SOAP Recibido.

Este mensaje de solicitud de ejecución de una operación en el servidor de Servicios Web, debe cumplir con las siguientes características:

- En caso de que se configure el servidor HTTP con autentificación básica, debe contener la clave y usuario para poder ser atendido.
- Debe indicar el Servicio Web al que pertenece ese mensaje SOAP.
- Indicar la operación que se esta solicitando será atendida.

La validación de las características antes mencionadas se realiza a través de los archivos de configuración de cada Servicio.

El mensaje con un resultado exitoso enviado al consumidor, se muestra en la figura 4.6. El elemento *todo:Resultado* contiene un documento en XML que puede ser transformado para presentar los resultados.

Mensaje SOAP de Respuesta

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <soapenv:Body>
        <todo:INTERPRETADO xmlns:todo="urn:xmethods-CurrencyExchange">
            <todo:Resultado>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&#xd;
&lt;Resultado_Operacion&gt;&lt;Res_Inst_0&gt;&lt;Registro&gt;&lt;IDDEPT&gt;&lt;![CDATA[10]]&gt;&lt;/IDDE
PT&gt;&lt;NOMBRE&gt;&lt;![CDATA[ACCOUNTING]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;Registro&gt;&l
t;IDDEPT&gt;&lt;![CDATA[20]]&gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;&lt;![CDATA[RESEARCH]]&gt;&lt;/NOMB
RE&gt;&lt;/Registro&gt;&lt;Registro&gt;&lt;IDDEPT&gt;&lt;![CDATA[30]]&gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;
&lt;![CDATA[SALES]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;Registro&gt;&lt;IDDEPT&gt;&lt;![CDATA[40]]&
gt;&lt;/IDDEPT&gt;&lt;NOMBRE&gt;&lt;![CDATA[OPERATIONS]]&gt;&lt;/NOMBRE&gt;&lt;/Registro&gt;&lt;/R
es_Inst_0&gt;&lt;/Resultado_Operacion&gt;</todo:Resultado>
        </todo:INTERPRETADO>
    </soapenv:Body>
</soapenv:Envelope>
```

Figura 4.6 Ejemplo de Mensaje SOAP Enviado.

Un mensaje SOAP que informa al usuario que ocurrió un error durante la atención de la solicitud enviada se muestra en la figura 4.7.

Mensaje SOAP con Error

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <soapenv:Body>
        <soapenv:Fault>
            <faultcode>soapenv:Error en la ejecucion de la operacion solicitada</faultcode>
            <faultstring>La fuente de Datos para el Servicio: SERVICIOORACLE no esta disponible por el
momento</faultstring>
            <faultactor>ERROR EN SERVIDOR</faultactor>
            <detail/>
        </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

Figura 4.7 Ejemplo de Mensaje SOAP con Información de Error.

Los métodos implementados para esta clase son los mostrados en la figura 4.8.

Método	Descripción
public MensajeSOAP(InputStream MensajeLeido)	En base a la cadena recibida por el consumidor crea una instancia de la clase MensajeSOAP.
public String getMetodo()	Obtiene el nombre de la operación solicitada para ejecución.
public String creaSOAP(String resultado)	En base a un documento XML como parámetro crea el mensaje SOAP de resultado.
public String creaErrorSOAP(String Errores, String actor)	Crea un mensaje SOAP de error que será enviado al consumidor.
public Vector getLisPara()	Obtiene una lista de los parámetros recibidos en el mensaje SOAP.

Figura 4.8 Lista de Métodos para la Clase MensajeSOAP.

A través de los métodos listados en la figura 4.8 se cubre el requerimiento de mensajes que deben existir entre consumidor y proveedor de Servicios Web.

4.1.4 Administración de API

Una de las características principales de esta aplicación es cargar los controladores JDBC, este módulo se encarga de proporcionar a la interfaz los elementos necesarios para abrir un archivo JAR y verificar si en el se encuentran controladores disponibles para acceso a base de datos.

Para poder cargar estos archivos, la aplicación ofrece al usuario una pantalla como la presentada en la figura 4.9, que muestra los controladores JDBC que se encuentran

disponibles y con la opción de agregar otros controladores. Una vez que estos son cargados en la aplicación (archivo de configuración JDBC), estarán disponibles cada vez que se reinicie la aplicación, cabe mencionar que los controladores cargados en línea, estarán disponibles desde el momento mismo de la carga, y no deberá reiniciarse el servidor para poder utilizarlo.

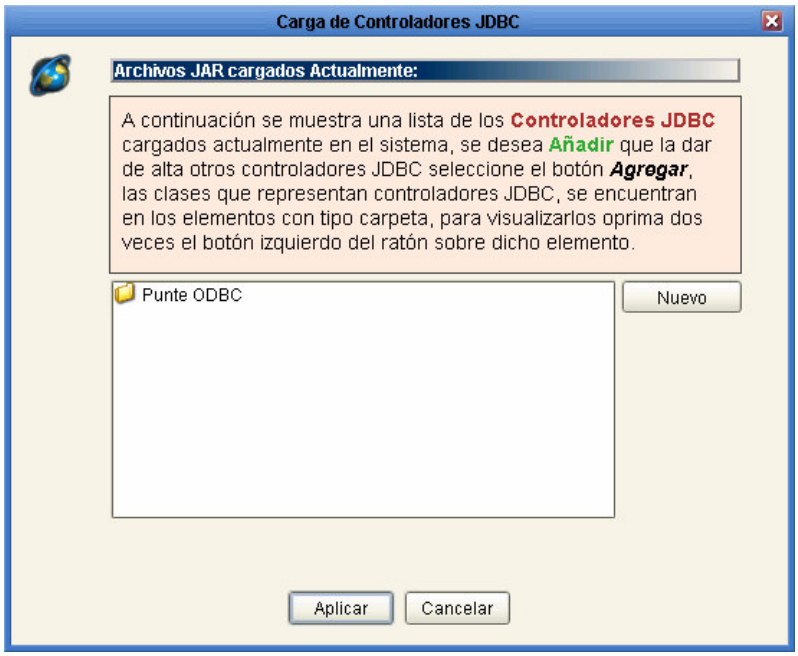


Figura 4.9 Pantalla para la Carga de Nuevos Controladores JDBC.

4.1.4.1 wsadmin.Util.CargarJAR

Esta clase fue desarrollada para poder brindar a la aplicación la posibilidad de cargar nuevos controladores JDBC en línea, esto es que no se necesite reiniciar la aplicación o el servidor para poderlo utilizar, con lo que se logra que sigan disponibles los Servicios Web ya configurados y que a la vez se puedan configurar nuevos con el controlador que acaba de ser cargado.

Los métodos principales de esta clase están descritos en la figura 4.10.

Método	Descripción
public void cargararchJAR(String arch)	Se encarga de cargar un archivo JAR especificado por su ruta completa.
public void cargararchJAR(String path, String arch, boolean agregvec, ProgressMonitor pm)	Se encarga de cargar un archivo JAR de acuerdo a su ruta y nombre, además de determinar si es necesario verificar si existen controladores JDBC.
public void CargarClases(String dir)	Realiza la carga de archivos JAR que se encuentren en el directorio especificado.
protected boolean yaCarg(URL archivo, URLClassLoader clases)	Verifica si un archivo JAR ya se encuentra cargado en la aplicación.
void ChecaDriver(String arch, ProgressMonitor pm)	Abre un archivo JAR para verificar las clases existentes y poder determinar cuales clases corresponden a controladores JDBC.
protected boolean esDriver(String clase, String NJar)	En base a la clase que recibe como parámetro revisa si es un controlador JDBC.
public boolean EjecutaMetodo(String clase, String metodo, Object[] arg)	Ejecuta un método de la clase que pasa como parámetro.

Figura 4.10 Lista de métodos para la Clase CargarJAR.

4.1.5 Control del Acceso a Base de Datos

Se construyó un módulo que permite de manera ordenada acceder al SABD elegido de acuerdo a la configuración de cada Servicio Web, sin embargo no es de uso exclusivo para la atención y configuración de los Servicios Web. Esta interfaz cuenta con un módulo donde el usuario puede acceder al SABD a través de la conexión JDBC previamente configurada.

4.1.5.1 wsadmin.BaseD.BaseDatos

Esta clase se encarga de llevar a cabo todas las peticiones necesarias al SABD a través de JDBC, permitiendo recuperar información propia del SABD además de enviar sentencias SQL.

Esta clase contiene métodos que le permiten saber la configuración de cada SABD, ya que las conexiones se realizan a través de JDBC, permitiendo a esta aplicación ser genérica y que la búsqueda de los metadatos no dependan de cada SABD, simplemente se accede a los métodos ofrecidos por JDBC y con estos se forman y recuperan los datos necesarios para permitir al usuario configurar los Servicios Web o para poder trabajar con la base de datos elegida.

Este módulo permite conocer como esta conformada la base de datos a la que se conecta cada Servicio Web, por lo que se cuenta con métodos que permiten recuperar los elementos que están contenidos en la base de datos y por cada elemento saber su composición, por ejemplo recuperar una lista de vistas, tablas, procedimientos

almacenados, entre otros y de estos elementos recuperar los campos y/o parámetros, así como los resultados arrojados por funciones y/o procedimientos almacenados.

Los métodos que cuenta esta clase se muestran en la figura 4.11.

Método	Descripción
public Vector getColumnas(ResultSet Resultados) throws Exception	Obtiene una lista de los campos que contiene el ResultSet que es pasado como parámetro.
public Vector getDatos(ResultSet Resultados) throws Exception	Convierte los datos de un ResultSet a Vector.
public Object[] ConsultaTab(String query)	Envía una sentencia SQL a la base de datos, regresando el resultado en forma de arreglo de objetos.
public void EjecutaDDLVista(String nombrevista, String query, boolean existe)	Crea una nueva vista en la base de datos.
public void EjecutaDDLProc(String query)	Crea un procedimiento almacenado y/o función en la base de datos.
public Vector ConsMetad()	Recupera las características generales de la base de datos a la cual se conecta.
public Vector Obtener_Tablas(String tipo)	Obtiene una lista de todas las tablas que en base a la configuración de la conexión JDBC puede acceder.
public Vector Obtener_Campos(String Tabla)	Obtiene una lista de los campos de la tabla especificada.
public Vector Obtener_Procs()	Obtiene una lista de todos los procedimientos y/o funciones que en base a la configuración de la conexión JDBC puede acceder.
public Vector Obtener_ProcPara(String Proc, int tipo)	Obtiene los parámetros del procedimiento y/o función especificada.
protected String TipoParam(int tipo)	Determina el nombre del tipo de parámetro.
public Vector getFunNum()	Obtiene una lista de las funciones numéricas de la base de datos.
public Vector getFunCad()	Obtiene una lista de las funciones para cadena de la base de datos.
public Vector getFunFec()	Obtiene una lista de las funciones para fechas de la base de datos.
public Vector getFunSis()	Obtiene una lista de las funciones de sistema de la base de datos.
public CallableStatement creaLlamProc(String sen)	Ejecuta una llamada a procedimiento y/o función.

Figura 4.11 Lista de Métodos para la Clase BaseDatos.

Existen métodos que envían sentencias SQL al SABD e interpretan los resultados que son arrojados por el SABD, con ello se informa los resultados al módulo que así lo haya solicitado, para posteriormente crear el XML de respuesta e informar al cliente o consumidor sobre los resultados de la ejecución de Servicios Web.

4.1.6 Servidor HTTP

Los Servicios Web deben tener un proveedor que permita a los consumidores poder conocer la composición de cada Servicio y solicitar su ejecución. Para que este proyecto no dependiera de otras aplicaciones extra se desarrolló un módulo que se encarga de escuchar peticiones, sin embargo este servidor de Servicios Web esta orientado específicamente a atender solicitudes que busquen descripción de algún Servicio Web o atiende las solicitudes que vengan como mensajes SOAP, y en caso de que la solicitud recibida no cumpla con alguna de las mencionadas, se enviará un mensaje de error.

4.1.6.1 wsadmin.http.ServHTTP

Para poder dar atención a las solicitudes de los consumidores, es necesario brindar un servidor que escuche dichas peticiones a través de un puerto TCP (configurable por el usuario) y que recibe peticiones HTTP, de acuerdo a la petición será las actividades que realice el servidor de Servicios Web.

Esta clase se encarga de iniciar o detener el servidor, además de crear los hilos de atención para cada consumidor.

El usuario podrá configurar el puerto, el número máximo de conexiones que podrá atender y el tiempo de espera de las solicitudes para cada Servicio Web. Con esta información se crea una instancia de esta clase para poder estar escuchando las solicitudes de los consumidores, si se realiza cambio en el puerto TCP seleccionado, las descripciones de los Servicios Web son actualizadas para que correspondan con la configuración actual del servidor.

Los métodos principales de esta clase se listan en la figura 4.12.

Método	Descripción
public ServHTTP(PServidor props, javax.swing.JTextArea EnvMen)	Se encarga de crear el servidor de Servicios Web.
public void run()	Proceso que escucha las peticiones de los consumidores.
public boolean AceptaConexion()	Determina si la petición es válida.
public void CerrarServ()	Detiene la ejecución del servidor.
public boolean getCorriendo()	Verifica si se encuentra corriendo el servidor de Servicios Web.

Figura 4.12 Lista de Métodos para la Clase ServHTTP.

Una característica importante de esta clase es que es ejecutada de manera concurrente a la aplicación global, esto es, que aunque este corriendo el servidor de Servicios Web, el usuario puede seguir creando nuevos Servicios y nuevos elementos en forma concurrente.

4.1.6.2 wsadmin.http.Hilo_Atencion

Al tener un módulo escuchando peticiones en un puerto TCP, cada vez que se realice una petición esta debe ser atendida, si se hiciera en forma serial, la atención de los Servicios Web sería muy lenta y podría ocasionar que la conexión entre el cliente y el servidor se perdiera, por lo que se desarrolló esta clase que a través del uso de Hilos da atención a cada una de las solicitudes en forma concurrente; cada instancia de esta clase se encarga de dar servicio a una solo petición.

Una vez que se crea una instancia esta se encarga de identificar el tipo de petición y de validar, en su caso, al usuario que la esta solicitando.

4.1.6.2.1 Solicitud de Descripción de Servicio Web (WSDL)

Cuando el consumidor solicita este tipo de petición el servidor debe:

- Verificar el tipo de servidor que se esta ejecutando, si es un servidor con autentificación básica debe verificar que se envíe el usuario y contraseña, si no es recibido se regresa al consumidor el mensaje de la figura 4.13.

Solicitud de Autentificación
HTTP/1.1 401 Authorization Required WWW-Authenticate: basic realm="Es necesario un Usuario y clave de acceso"

Figura 4.13 Mensaje para Solicitar Autentificación al Consumidor.

- Si se recibe correctamente el usuario o no es un servidor con autentificación, se busca el archivo solicitado; si es un archivo válido se envía al consumidor.
- Cuando no se encuentra el archivo solicitado se envía el mensaje de la figura 4.14.

Recurso no encontrado

```
<HTML>
<HEAD>
    <TITLE>El archivo solicitado no fue
encontrado</TITLE>
</HEAD>
<BODY>
<div style="background-color: #008000">
  <p align="center"><font face="Arial Black" size="5"
color="#FFFFFF">ERROR:</font><font color="#FFFF00"
face="Arial Black" size="5">
  404<br>
  </font><font face="Arial Black" size="5"
color="#FFFFFF">
  RECURSO NO DISPONIBLE</font></div>
<p><font face="Arial" size="4"><b>Los Errores
Generados pueden ser:</b><br>Este servidor no cuenta
con el recurso solicitado.<br>No es un recurso válido
```

```
(Solo atiende peticiones de Servicios Web, WSDL y/o
SOAP)</font></p>
<div style="border-top-style: ridge; border-top-
width: 3">
  <i><font face="Arial" size="1"
color="#A0A0A0">Servidor de Servicios
Web</font></i></div>
</BODY>
</HTML>
```

Figura 4.14 Mensaje de Recurso no Encontrado.

- Si el archivo solicitado no está permitido o no se cumplió con la autentificación, se envía el mensaje de la figura 4.14.

Sin Permisos

```
<HTML>
<HEAD>
    <TITLE>El archivo solicitado no fue
encontrado</TITLE>
</HEAD>
<BODY>
<div style="background-color: #008000">
  <p align="center"><font face="Arial Black" size="5"
color="#FFFFFF">ERROR:</font><font color="#FFFF00"
face="Arial Black" size="5">
  401<br>
  </font><font face="Arial Black" size="5"
color="#FFFFFF">
  NO TIENE PERMISOS PARA INGRESAR A ESTA
PÁGINA</font></div>
<p><font face="Arial" size="4">Verifique el usuario y
contraseña proporcionados</font></p>
<div style="border-top-style: ridge; border-top-
width: 3">
  <i><font face="Arial" size="1"
color="#A0A0A0">Servidor de Servicios
Web</font></i></div>
</BODY>
</HTML>
```

Figura 4.15 Mensaje de que el Consumidor no Cuenta con Permisos para Acceder al Recurso.

4.1.6.2.2 Solicitud de Ejecución de Servicio (Mensaje SOAP)

Por otro lado, el consumidor puede enviar una petición de ejecución de alguna operación de los Servicios Web, lo que provoca lo siguiente:

- Se valida que exista el Servicio solicitado.

- Si existe este Servicio recuperar su configuración y la configuración de la operación solicitada.
- Se identifica la operación que se desea ejecutar.
- Se compara la configuración con la información recibida en el mensaje SOAP.
- Si la solicitud de ejecución es válida, se crea una instancia de la clase *wsadmin.http.AteServicio* para ejecutar la operación.
- En caso de que no se presente algún error en cualquier parte de la atención a esta petición, se enviará un mensaje SOAP comunicando el resultado de la ejecución.
- En caso de que se presente un error en cualquier parte de la ejecución, se creará un mensaje SOAP con la información del error.

Para poder enviar y poder interpretar los mensajes recibidos se sigue la especificación del protocolo HTTP [L-38].

4.2 Archivos de Configuración

El desarrollo de la aplicación cumple con los requisitos que se presentaron en capítulos anteriores, sin embargo debe guardarse toda la información que en ella se configura y aunque existen una gran variedad de formatos entre los que se pudieron elegir para guardar la información, incluso crear uno propio, se decidió seguir con la misma línea de aplicaciones semejantes y utilizar XML como formato para crear los archivos de configuración.

En esta sección se describe cada uno de los archivos de configuración, mostrando los elementos de cada uno y su composición, así como los valores permitidos para cada atributo o elemento.

4.2.1 Elementos del archivo de Configuración del Servidor

El archivo principal es llamado *Conf.xml* y se encuentra en el mismo directorio que el archivo que ejecuta la aplicación. Este archivo esta compuesto según el diagrama presentado en la figura 4.16 y un ejemplo del mismo es presentado en la figura 4.17.

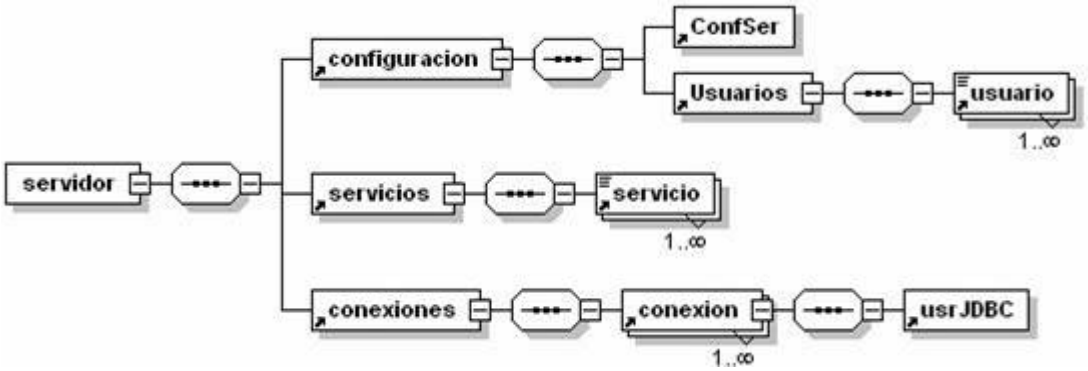


Figura 4.16 Estructura del Archivo Conf.xml.

Servicio Web para ORACLE configurado (conf.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<servidor>
    <configuracion>
        <ConfSer nombre="SSWdataBase" dirIP="192.168.184.1" puerto="1010" tipo="HTTP con
            Autentificación" directorio="D:\Servidor\ConDatos" MaxCon="607" TiempoSesion="0"/>
        <Usuarios>
            <usuario nombre="Luis González" usrID="luisge" clave="bHVpc2dlOnBhc28=" tipo="ADM">Luis
                González</usuario>
            <usuario nombre="usuario web Service" usrID="USUSERV" clave="VVNVU0VSVjpwYXNv"
                tipo="VIE"/>
        </Usuarios>
    </configuracion>
    <servicios>
        <servicio nombre="ServicioOracle" dirBase="/ServicioOracle.wsdb" archConf="ServicioOracle.xml">
<![CDATA[Descripción del Servicio]]></servicio>
    </servicios>
    <conexiones>
        <conexion nombre="ORAJDBC" url="jdbc:oracle:thin:@localhost:1521:BDD"
clase="oracle.jdbc.driver.OracleDriver" SerVis="select text from all_views where view_name=?"
SenPro="select text from all_source where name = ?">
            <usrJDBC usrID="scott" clave="c2NvdHQ6dGlnZXI="/>
        </conexion>
    </conexiones>
</servidor>
```

Figura 4.17 Estado del Archivo de Configuración Conf.xml Después de Configurar Algunos Elementos.

El elemento *configuracion* contiene la información de configuración del servidor de Servicios Web, esta información se organiza en los siguientes elementos:

ConfSer

Contiene los datos de la configuración sobre la cual se debe iniciar el servidor de Servicios Web para escuchar peticiones.

Usuarios

Contiene elementos Usuario que definen a cada uno de los usuarios, que en caso de que sea configurado el servidor de Servicios Web con autentificación, podrán hacer uso de los Servicios. Además define el usuario administrador de la aplicación, inicialmente sólo existe el usuario administrador:

- **Nombre:** nombre del usuario.
- **usrID:** nombre o alias con el cual será identificado el usuario.
- **clave:** clave de acceso a los Servicios Web, (esta clave esta codificada en BASE64, según la especificación de HTTP).
- **tipo:** determina el tipo de usuario, puede ser de Servicios Web o administrador.

Servicios

Este elemento inicialmente se encuentra vacío hasta que se configuren Servicios Web. Una vez que el usuario empieza a definir Servicios Web, se agregarán a este elemento hijos llamados Servicio que contendrán la información necesaria para poder identificar cada Servicio Web y formar su contenido:

- **Nombre:** identificador único para cada Servicio Web.
- **dirBase:** define como será identificado el Servicio en el momento que se solicite una ejecución de sus operaciones, y esta formado de la siguiente forma:

"/"+nombre+".wsdb"

- **archConf:** indica el nombre del archivo donde se encuentra el detalle para el Servicio que hace referencia este elemento, este nombre está conformado de la siguiente forma:

nombre+".xml"

Adicionalmente cada elemento *Servicio* contiene como valor una sección CDATA donde el usuario podrá escribir una breve descripción sobre el servicio configurado y que será parte del archivo WSDL del Servicio Web.

Conexiones

Como se mencionó esta aplicación esta orientada a la publicación de Servicios Web con acceso a base de datos, es por ello que deben configurarse conexiones JDBC y asignarlas a los Servicios para poder acceder a los SABD.

El elemento *Conexiones* se encarga de definir a cada una de estas conexiones JDBC, conteniendo subelementos *Conexion* que representan la configuración de cada JDBC:

- **Nombre:** identifica de manera única el JDBC configurado.
- **url:** contiene la cadena de conexión necesaria para acceder al SABD.

- **clase:** determina la clase Java del controlador a utilizar para realizar la conexión.
- **SerVis:** contiene una sentencia SQL que permite a la aplicación buscar la definición de una vista
- **SenPro:** contiene una sentencia SQL que permite a la aplicación buscar el código de una función o procedimiento almacenado
- **Elemento usrJDBC:** este elemento se encarga de la información del usuario que se conectará a la base de datos, contiene el usrID y la clave.

4.2.2 Elementos del archivo de Configuración de Servicio Web

En la sección 4.2.1 se presentó el archivo de configuración global del servidor de Servicio Web, esta sección presenta el detalle para cada servicio configurado, este archivo XML existe para cada Servicio Web, en el directorio raíz de la aplicación se podrán ver tantos archivos de configuración como Servicios Web configurados.

El nombre del archivo se compone con el nombre del Servicio y se encuentra en el mismo directorio que el archivo que ejecuta la aplicación, este archivo esta compuesto según el diagrama presentado en la figura 4.18.

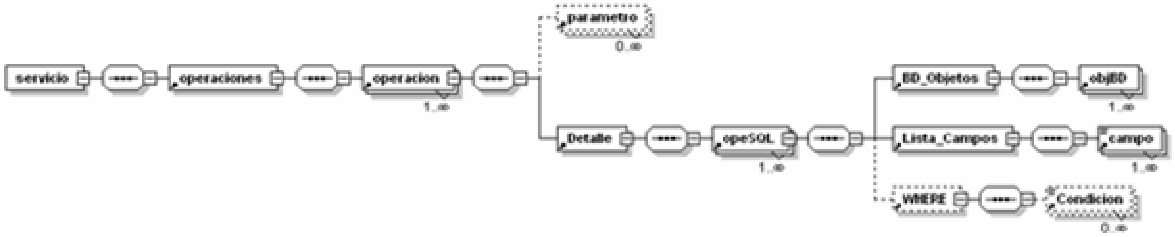


Figura 4.18 Estructura del Archivo de Configuración para Cada Servicio.

En las secciones siguientes se muestra cada uno de los elementos que contiene este archivo de configuración.

El elemento *Servicio* representa la raíz del documento, y presenta algunos atributos que representan la configuración a detalle para el Servicio al cual pertenece:

- **Nombre:** indica el Servicio Web al cual pertenece este archivo.
- **archWSDL:** representa la ruta para obtener el archivo que describe a este Servicio.
- **JDBC:** contiene la referencia a una conexión JDBC que podrá utilizar el Servicio Web para configurar sus operaciones.

El elemento *Operaciones* contiene cada una de las operaciones que componen al Servicio Web, así como sus detalles.

La información que presenta este elemento se decidió poner en este archivo y no en el archivo global de configuración, debido a que se buscó concentrar en un solo archivo la configuración a detalle de cada Servicio Web.

En elemento *Operacion* se concentra la información de cada operación que forma parte del Servicio Web; es un elemento sin atributos y sin valores y sirve para concentrar elementos *Operacion* con el siguiente detalle:

- **Nombre:** indica el nombre con el que será identificada la operación, tanto en el documento XML como en el archivo que describe al Servicio Web.

La figura 4.19 muestra algunos ejemplos de la configuración de operaciones, en donde la primera columna representa el texto XML de la operación, y la columna de la derecha representa la sentencia SQL a la cual se traduce.

XML	SQL
<Detalle> <opeSQL nombre="Inst0" tipo="S" retresult="1"> <BD_Objetos> <objBD nombre="EMP"/> <objBD nombre="DEPT"/> </BD_Objetos> <Lista_Campos> <campo nombre="idEmp" group_by="0" order_by="1"><![CDATA[EMP.EMPNO]]></campo> <campo nombre="Nombre" group_by="0" order_by="0"><![CDATA[EMP.ENAME]]></campo> <campo nombre="FechCont" group_by="0" order_by="0"><![CDATA[EMP.HIREDATE]]></campo> <campo nombre="Salario" group_by="0" order_by="0"><![CDATA[EMP.SAL]]></campo> <campo nombre="NomDep" group_by="0" order_by="0"><![CDATA[DEPT.DNAME]]></campo> <campo nombre="Puesto" group_by="0" order_by="0"><![CDATA[EMP.JOB]]></campo> </Lista_Campos> <WHERE> <Condicion tipo="C" operador="0" negada="0"><![CDATA[EMP.DEPTNO = DEPT.DEPTNO]]></Condicion> </WHERE> </opeSQL> </Detalle>	SELECT emp.empno idemp, emp.ename nombre, emp.hiredate fechcont, emp.sal salario, dep.dname nomdep, emp.job puesto FROM emp, dept WHERE emp.deptno = dept.deptno
<Detalle> <opeSQL nombre="Inst1" tipo="P" retresult="1" tipop="0" formejec="SCOTT.GETDATOSPOREMP"> <BD_Objetos> <objBD nombre="GETDATOSPOREMP"/> </BD_Objetos> <Lista_Campos> <campo nombre="GETDATOSPOREMP.IDEMP" tipo="IN-3"><![CDATA[idEmp]]></campo> <campo nombre="GETDATOSPOREMP.NOMBRE" tipo="OUT-12"><![CDATA[Regresa Valor]]></campo> <campo nombre="GETDATOSPOREMP.PUESTO" tipo="OUT-12"><![CDATA[Regresa Valor]]></campo> <campo nombre="GETDATOSPOREMP.FECHA" tipo="OUT-93"><![CDATA[Regresa Valor]]></campo> <campo nombre="GETDATOSPOREMP.SALARIO" tipo="OUT-3"><![CDATA[Regresa Valor]]></campo> <campo nombre="GETDATOSPOREMP.NAMEDEPTO" tipo="OUT-12"><![CDATA[Regresa Valor]]></campo> </Lista_Campos> </opeSQL> </Detalle>	**Call scott.getdatosporemp(?,?,?,?,?,?)** De acuerdo a la configuración el primer ? será sustituido por el parámetro recibido en el Servicio Web, el resto de los ? serán datos que regresará el procedimiento almacenado.

Figura 4.19 Traducción de XML a SQL.

El elemento *detalle* esta contenido dentro del elemento *Operacion* y se encarga de definir donde comienza el cuerpo de cada operación de cada Servicio Web.

Contiene al elemento *opeSQL* con la información necesaria para poder crear sentencias SQL, este elemento contiene los siguientes atributos:

- **nombre:** un identificador único para cada operación, este nombre se puede repetir en cuando cambie el nombre de la operación.
- **tipo:** indica el tipo de sentencia SQL que debe ser construida y, en base a ella, como serán interpretados los elementos hijos de este elemento, los tipos válidos para este elemento son:
    - **P:** para procedimiento.
    - **S:** para sentencia SQL de consulta.
    - **I:** para sentencia SQL de insertar datos en una tabla.
    - **U:** sentencia SQL para actualizar los datos de una tabla.
    - **D:** para eliminar datos de una tabla.
- **retresult:** indica a la aplicación si deberá crear un documento XML para informar sobre los resultados arrojados al ejecutar esta sentencia SQL. Lo valores son 0 (cero) en caso de que no se requiere informar sobre el resultado y 1 (uno) para indicar que se debe construir el documento XML.
- **tipoop:** indica si es un procedimiento o función.
- **formejec:** indica la forma en que debe ser invocado el procedimiento y/o función.

El elemento *BD_Objetos* informa a la aplicación cuales son los elementos de la base de datos que deberán tomarse en cuenta para formar la sentencia SQL, debe estar presente al menos un elemento *objBD*, en caso contrario no será una sentencia válida.

El elemento *objBD* indica a la aplicación el nombre del elemento de la base de datos que deberá formar parte de la sentencia SQL; este elemento podrá ser una tabla, vista, procedimiento almacenado ó una función, y dependerá de la configuración del elemento *opeSQL*.

Lista_Campos es un elemento opcional y dependerá de la configuración del elemento *opeSQL*, ya que si la operación representa la ejecución de una función o de un procedimiento almacenado, puede no tener parámetros, por lo que este elemento no estaría presente en el documento XML de configuración del Servicio. Si se encuentra presente deberá tener al menos un elemento campo, el cual representa un parámetro o el campo de una sentencia de consulta o actualización en la sentencia SQL final, sus atributos son:

- **nombre:** dependiendo del tipo de instrucción, representará el nombre que identifica al campo en la base de datos o el nombre con el que se conocerá al campo una vez ejecutada la sentencia SQL.

- **tipo:** indica el tipo de dato JDBC que representa este campo, además de informar a la aplicación si es un campo de entrada o salida; los campos que sean de salida (RET ó OUT) podrán ser incluidos en el documento XML de respuesta. Este atributo estará presente cuando la instrucción a ejecutar sea la llamada a un procedimiento almacenado y/o función.
- **group_by:** estará presente cuando se trate de una sentencia SQL de consulta e indica si el campo forma parte de la cláusula "group by".
- **order_by:** estará presente cuando se trate de una sentencia SQL de consulta e indica si el campo forma parte de la cláusula "order by".

Además los elementos tipo campo, podrán contener una sección CDATA para indicar como estará compuesta la sentencia SQL, esto es:

- **Consulta:** Representa la expresión que deberá ser consultada para regresar en este campo.
- **Insertar y/o Actualizar:** representa la expresión con la que será actualizada el campo al que se hace referencia.
- **Procedimiento y/o Función:** representa el parámetro del Servicio Web que será puesto en el lugar del campo que hace referencia.

El elemento *Where* estará presente en sentencias que no ejecuten algún procedimiento almacenado o función, y en las sentencias que así lo necesiten (SELECT, UPDATE y DELETE).

Contiene un árbol de elementos que representan las condiciones a crear en el momento de traducirlas a sentencias SQL. Estas condiciones están representadas por los elementos *Condicion*, y a su vez este elemento puede tener uno o más elementos *Condicion*. Cuando el elemento *Condicion* tiene hijos se toma como un grupo de condiciones, mientras que en el caso de no tener hijos se tomará como una condición final.

Los atributos del elemento *Condicion* son:

- **tipo:** indica si es una condición (C) o un grupo de condiciones (G).
- **operador:** indica el tipo de operador lógico con el que será concatenada esta condición o grupo de condiciones, 0 (cero) representa un AND y 1 (uno) representa el operador OR.
- **negada:** indica si deberá formarse esta condición con un operador de negación.

4.2.3 Elementos del archivo de Configuración Auxiliar JDBC

Existe un tercer archivo de configuración dónde es almacenada información sobre los controladores JDBC, este archivo contiene información adicional que no es controlada por la aplicación.

El diagrama presentado en la figura 4.20 representa la composición del archivo XML para JDBC, su función es brindar algunas ayudas al momento de configurar conexiones JDBC y también cuando el usuario crea procedimientos o funciones con esta aplicación.

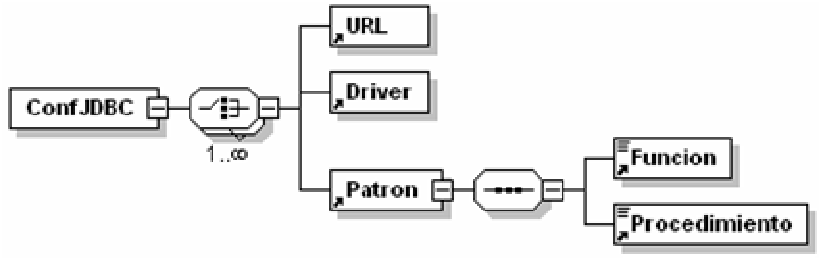


Figura 4.20 Estructura del Archivo de Configuración ConfJDBC.xml.

Los elementos que brindan una ayuda al momento de utilizar la aplicación son:

- **URL:** define la cadena de conexión a partir de una clase seleccionada; con esto si el usuario utiliza una clase que se encuentre en alguna URL ya definida la aplicación mostrará un patrón del URL a utilizar, los atributos son:
    - **clase:** indica la clase Java del controlador JDBC que debe utilizarse con este URL.
    - **url:** indica el patrón de cadena de conexión para dicho controlador JDBC.
- **Patron:** este elemento le indica a la aplicación en base al tipo de SABD, cuales son los patrones o plantillas para crear un procedimiento almacenado o una función. Contiene dos elementos:
    - **Funcion:** se pone la sintaxis que se debe utilizar para crear una función.
    - **Procedimiento:** se pone la sintaxis que se debe utilizar para crear un procedimiento almacenado.

Estos elementos permiten que se definan algunas opciones que son comúnmente utilizadas durante la configuración de los Servicios Web, simplificando al usuario al momento de crearlos.

4.2.3.1 Elemento DRIVER

El único elemento controlado por la aplicación es el elemento *Driver*, que define los archivos que contienen los controladores JDBC y que serán cargados durante el inicio de la aplicación.

Cuando el usuario incluye un nuevo controlador en la aplicación este es almacenado bajo un elemento *Driver*, permaneciendo disponible hasta que se elimine de esta configuración o no se encuentre disponible el controlador.

Los atributos que maneja este elemento son:

- **path:** indica la ruta completa donde se localiza el controlador JDBC.
- **estado:** indica si el controlador esta disponible o no, si el estado contiene un valor de 0 (cero) el controlador JDBC esta disponible, en caso de que al momento de realizar la carga de este controlador ocurra algún error, su estado cambiará a 1 (uno) indicándole a la aplicación que no deberá cargarse este archivo.

La figura 4.9 muestra la pantalla de carga de los controladores y los elementos *Driver* de este archivo de configuración. Algunos elementos *Driver* tienen su atributo estado en 1 (uno), debido a esto no aparecen en la pantalla de Controladores JDBC disponibles.

La figura 4.21 muestra el archivo *Conf.xml* con la configuración del Servicio Web para acceder al SABD ORACLE, este Servicio es llamado *ServicioOracle* y esta ligado a la conexión JDBC llamada ORAJDBC.

Servicio Web para ORACLE configurado (conf.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<servidor>
    <configuracion>
        <ConfSer nombre="SSWdataBase" dirIP="192.168.184.1" puerto="1010" tipo="HTTP con
            Autentificación" directorio="D:\Servidor\ConDatos" MaxCon="607" TiempoSesion="0"/>
        <Usuarios>
            <usuario nombre="Luis González" usrID="luisge" clave="bHVpc2dlOnBhc28=" tipo="ADM">Luis
                González</usuario>
            <usuario nombre="usuario web Service" usrID="USUSERV" clave="VVNVU0VSVjpwYXNv"
                tipo="VIE"/>
        </Usuarios>
    </configuracion>
    <servicios>
        <servicio nombre="ServicioOracle" dirBase="/ServicioOracle.wsdb" archConf="ServicioOracle.xml">
<![CDATA[Descripción del Servicio]]></servicio>
    </servicios>
    <conexiones>
        <conexion nombre="ORAJDBC" url="jdbc:oracle:thin:@localhost:1521:BDD"
clase="oracle.jdbc.driver.OracleDriver" SerVis="select text from all_views where view_name=?"
SenPro="select text from all_source where name = ?">
            <usrJDBC usrID="scott" clave="c2NvdHQ6dGlnZXI="/>
        </conexion>
    </conexiones>
</servidor>
```

Figura 4.21 Estado del Archivo de Configuración Conf.xml Después de Configurar Algunos Elementos.

El archivo de configuración específico para el Servicio Web *ServicioOracle* es el que se muestra en la figura 4.22.

Servicio Web para ORACLE configurado (ServicioOracle.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<servicio nombre="ServicioOracle" archWSDL="ServicioOracle.wsdl" JDBC="ORAJDBC">
    <operaciones>
        <operacion nombre="GetEmpleados">
            <Detalle>
                <opeSQL nombre="Inst0" tipo="S" retresult="1">
                    <BD_Objetos>
                        <objBD nombre="EMP"/>
                        <objBD nombre="DEPT"/>
                    </BD_Objetos>
                    <Lista_Campos>
                        <campo nombre="idEmp" group_by="0"
order_by="1"><![CDATA[EMP.EMPNO]]></campo>
                        <campo nombre="Nombre" group_by="0"
order_by="0"><![CDATA[EMP.ENAME]]></campo>
                        <campo nombre="FechCont" group_by="0"
order_by="0"><![CDATA[EMP.HIREDATE]]></campo>
                        <campo nombre="Salario" group_by="0"
order_by="0"><![CDATA[EMP.SAL]]></campo>
                        <campo nombre="NomDep" group_by="0" order_by="0"><![CDATA[ DEPT.DNAME
]]></campo>
                        <campo nombre="Puesto" group_by="0" order_by="0"><![CDATA[EMP.JOB
]]></campo>
                    </Lista_Campos>
                    <WHERE>
                        <Condicion tipo="C" operador="0" negada="0"><![CDATA[ EMP.DEPTNO  =
DEPT.DEPTNO  ]]></Condicion>
                    </WHERE>
                </opeSQL>
            </Detalle>
        </operacion>
        <operacion nombre="getDatporEmp">
            <parametro nombre="idEmp" tipo="int" longd="4" longe="0"/>
            <Detalle>
                <opeSQL nombre="Inst1" tipo="P" retresult="1" tipop="0"
formejec="SCOTT.GETDATOSPOREMP">
                    <BD_Objetos>
                        <objBD nombre="GETDATOSPOREMP"/>
                    </BD_Objetos>
                    <Lista_Campos>
                        <campo nombre="GETDATOSPOREMP.IDEMP" tipo="IN-3"><![CDATA[ idEmp
]]></campo>
                        <campo nombre="GETDATOSPOREMP.NOMBRE" tipo="OUT-
12"><![CDATA[Regresa Valor]]></campo>
                        <campo nombre="GETDATOSPOREMP.PUESTO" tipo="OUT-
12"><![CDATA[Regresa Valor]]></campo>
                        <campo nombre="GETDATOSPOREMP.FECHA" tipo="OUT-
93"><![CDATA[Regresa Valor]]></campo>
                        <campo nombre="GETDATOSPOREMP.SALARIO" tipo="OUT-
3"><![CDATA[Regresa Valor]]></campo>
                        <campo nombre="GETDATOSPOREMP.NAMEDEPTO" tipo="OUT-
12"><![CDATA[Regresa Valor]]></campo>
                    </Lista_Campos>
                </opeSQL>
            </Detalle>
        </operacion>
    </operaciones>
</servicio>
```

Figura 4.22 Archivo de Configuración del Servicio ServicioOracle.

En la figura 4.22 podemos observar que se encuentran definidas dos operaciones para este Servicio:

- **GetEmpleados:** que contiene un elemento *opeSQL* de tipo consulta (SELECT).
- **getDatporEmp:** que realiza un llamado a un procedimiento almacenado.

Estos archivos crecerán según se configuren servicios y conexiones dentro de la aplicación que este trabajo presenta.

4.3 Software de Soporte para el Desarrollo de la Aplicación

Para el desarrollo de la aplicación se seleccionaron algunas librerías que apoyan la orientación de este trabajo, las mostradas en la figura 4.23 fueron seleccionadas por su implementación y por no depender de una plataforma en específico, se mencionan también software empleado para la construcción de este trabajo.

Campo	Descripción
AXIS	Evolución del APACHE SOAP, que de manera simple permite publicar clases de Java como Servicios Web. Esta aplicación contiene la mayor parte de las librerías utilizadas para el desarrollo de esta aplicación, aunque no todas fueron desarrolladas por el mismo creador de Axis. Las librerías que Axis contiene y que se utilizaron para el desarrollo de este proyecto son principalmente para el manejo de mensajes SOAP y para la generación de documentos WSDL.
XPATH	Se utiliza en conjunto con XML Dom para realizar búsquedas de elementos en documentos XML, con este tipo de búsquedas XPATH permite recuperar uno o varios elementos a la vez. Los parámetros de búsqueda pueden ser en base a los valores de un elemento o en base a sus atributos.
WSDL4J	Es un API que permite leer y generar documentos WSDL en base a los elementos que lo conforman, esta librería se eligió por que sus métodos están ligados a cada elemento que debe existir en el documento WSDL, además cumple con la especificación de WSDL, por otro lado es una librería que utiliza Axis para la generación de los documentos WSDL.

JBuilder	IDE utilizado para la construcción de la aplicación, principalmente por las características ofrecidas que van ligadas con la orientación de este trabajo: Creación de interfaces gráficas. Interacción con Servicios Web.

Figura 4.23 Software de Soporte.

4.4 Resumen

En este capítulo se presentaron las principales clases que llevan el control de la aplicación, además de mostrar la estructura de los archivos de configuración y entender que es posible modificarlo de manera manual siguiendo la estructura y reglas de cada uno. Por otro lado se presentaron las herramientas utilizadas en el desarrollo de la aplicación.

Capítulo V
Pruebas

En los capítulos anteriores se presentó el proceso de desarrollo de la aplicación. Una vez que se tiene completa la aplicación y funcionando, debe ser puesta a prueba en la mayoría de los módulos que la componen, principalmente en los módulos de mayor carga, como son la creación y la publicación de los Servicios Web.

Los archivos de configuración iniciales y finales de las pruebas de esta aplicación están contenidos en el CD que acompaña a este trabajo, y cuyo contenido se presenta en el anexo F.

5.1 Establecimiento del Entorno de Pruebas

Es necesario contar con un entorno donde se llevan a cabo las pruebas de la aplicación, así como contar con herramientas diversas que permitan revisar el comportamiento, las tareas a realizar para poder probar cada parte de la aplicación.

Como se mencionó anteriormente la tarea principal de la aplicación es crear y publicar Servicios Web para base de datos, sin embargo para poder probar esta funcionalidad es necesario realizar una serie de pasos antes de poderlos publicar:

- Elección de los SABD que serán utilizados para la realización de las pruebas, para probar la interoperabilidad de la aplicación.
- Creación de la base de datos.
- Definición de los Servicios Web que deberán ser creados.
- Creación de las operaciones para cada Servicio Web de acuerdo a las opciones establecidas por consumidor de los mismos.
- Definir las aplicaciones cliente que serán utilizadas para probar al proveedor de Servicios Web.

En base a los puntos anteriores se podrán realizar diversas pruebas de funcionalidad y operabilidad. El SABD elegido para realizar estas pruebas de comparación con otras herramientas similares fue Oracle 9i, ya que entre las características que presenta Oracle es la de crear XML a partir de una sentencia SQL.

Las pruebas fueron realizadas de acuerdo a la orientación del trabajo, por lo que el entorno de pruebas fue enfocado a crear Servicios Web con acceso a base de datos y que arrojaran resultados en XML. En base a lo anterior se agruparon de la siguiente forma:

- Necesidades de infraestructura para publicar Servicios Web.
- Conocimientos necesarios para su utilización.
- Tiempo de respuesta de cada opción.

Por otro lado se realizaron diversas pruebas a esta aplicación donde se involucraron a los SABD MySQL, PostgresSQL y Oracle para publicar Servicios Web similares con diferentes fuentes de datos.

5.1.1 Creación de la Base de Datos en los SABD elegidos

Para poder utilizar esta herramienta el recurso principal es contar con un sistema administrador de base de datos. Aunque esta aplicación contiene un módulo para interactuar con el SABD en forma directa, se recomienda contar con una base de datos ya creada en el mismo.

La figura 5.1 muestra el diagrama Entidad-Relación de la base de datos diseñada para la realización de pruebas a la aplicación.

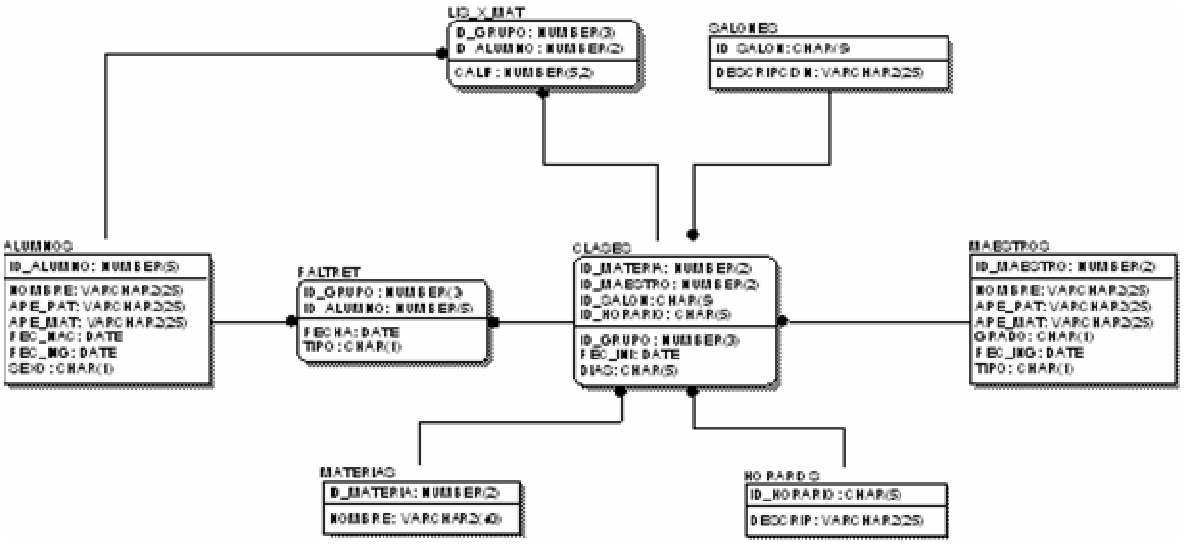


Figura 5.1 Diagrama Entidad-Relación.

Este esquema corresponde a un ejemplo sobre la información de una escuela y que esta bajo responsabilidad del departamento de control escolar, contiene información sobre alumnos, profesores, materias y grupos que serán impartidos.

Estos elementos pueden ser creados desde el SABD o a través de esta aplicación. El script de sentencias SQL para crear la base de datos en forma completa, de acuerdo a cada sistema administrador de base de datos, se presenta en el anexo C. La descripción de cada tabla es presentada en la figura 5.2, y en ella se observa el nombre de la tabla, los campos que contiene y la descripción de la misma.

Tabla	Descripción
ALUMNOS ID_ALUMNO: NUMBER(5) NOMBRE: VARCHAR2(25) APE_PAT: VARCHAR2(25) APE_MAT: VARCHAR2(25) FEC_NAC: DATE FEC_ING: DATE SEXO: CHAR(1)	Esta tabla se encarga de almacenar los datos específicos por cada alumno: • Id_Alumno: Clave del alumno. (PK) • Nombre: Nombre del alumno • Ape_pat: Apellido paterno. • Ape_pat: Apellido materno. • Fec_nac: Fecha de nacimiento. • Fec_ing: Fecha de ingreso. • Sexo: Si es hombre o Mujer.
MAESTROS ID_MAESTRO: NUMBER(2) NOMBRE: VARCHAR2(25) APE_PAT: VARCHAR2(25) APE_MAT: VARCHAR2(25) GRADO: CHAR(1) FEC_ING: DATE TIPO: CHAR(1)	Contiene la información de cada Maestro: • Id_maestro: Clave del maestro. • Nombre: Nombre del alumno • Ape_pat: Apellido paterno. • Ape_pat: Apellido materno. • Grado: Máximo grado de estudios. • Fec_ing: Fecha de ingreso. • Tipo: Profesor de tiempo completo o parcial.

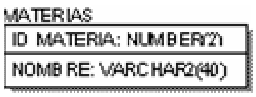
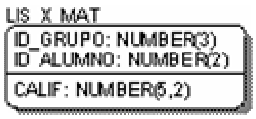
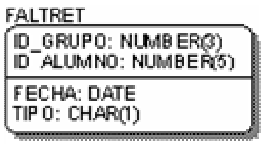
	Contiene la información de todas las materias que se contemplan impartir.
	Información de los horarios que se manejan para impartir clases.
	Información sobre los salones disponibles.
	Contiene información sobre que alumnos están inscritos en que materias y las calificaciones obtenidas.
	Indica la información de las clases que están disponibles, en base a grupo, maestro, materia, horario y salón, además de una fecha de apertura y los días de la semana en que es impartida (LMmJV).
	Registra la información de retardos y faltas por alumno y materia.

Figura 5.2 Tablas.

5.1.2 Equipo Utilizado

Las pruebas realizadas a esta aplicación estuvieron soportadas por la siguiente configuración de equipo:

- Procesador Pentium IV a 2.4 Mhz.
- Memoria Ram de 256 Mb.
- Disco Duro de 80 Gb (50 Gb disponibles).
- Sistema Operativo (Windows 2000 Server y Linux Red Hat 9.0).
- Oracle 9i release 2 (Servidor y Cliente).
- Tomcat 5.0.
- Java Runtime Environment 1.4_02.

5.2 Comparativa con Herramientas Similares

Una aplicación orientada a la publicación de Servicios Web debe ser comparada con herramientas que ofrecen funcionalidades similares, por lo que se comparó con:

- Aplicación de propósito general para la publicación de Servicios Web (Axis).
- Aplicación para publicar Servicios Web con acceso a Base de Datos (Virtuoso).
- Aplicación que a través de un URL genere documentos XML a partir de una fuente de datos (Servlets).

Virtuoso no puedo ser comparada al 100% debido a que es una herramienta con licencia y que no permite trabajar con todas sus funciones.

5.2.1 Necesidades de infraestructura para publicar Servicios Web

La primera prueba realizada fue determinar en base a la documentación de cada herramienta los requerimientos de equipamiento que debe contemplar el usuario de la misma. Cabe mencionar que las pruebas realizadas son del lado del proveedor, no del consumidor, por lo que el resultado final de estas pruebas es un documento XML. El consumidor lo hará a través de una dirección de Internet y un navegador web, la figura 5.3 es una tabla comparativa entre estas aplicaciones.

Virtuoso	Axis	Servlets
• Licencia para activarla. • Librerías para generar XML.	• Servidor de Aplicaciones (Java). • JDK versión 1.4 o superior. • Librerías para manipular XML. • Librerías para generar Servicios Web.	• Servidor de Aplicaciones con soporte para Servlets. • Librerías para generar XML.

Figura 5.3 Requerimientos de Instalación.

Como se observan en forma general los requerimientos, tales como un contenedor que soporte su tecnología, que en muchos casos depende de la plataforma a utilizar y presenta aún más requerimientos que los mostrados, además que en su mayoría dependen de herramientas de terceros para poder trabajar en forma eficiente.

Este trabajo tuvo como objetivo ser una interfaz simple y de fácil administración, por lo que las necesidades para utilizar esta aplicación son los siguientes:

- Java Runtime Environment 1.4x o superior.

- Espacio disponible en disco de 15 Mb.

Con lo que se observa un nivel de requerimientos inferior a las demás aplicaciones.

5.2.2 Conocimientos Necesarios para su utilización

Otro punto de comparación importante es el de verificar las necesidades de cada aplicación en base al conocimiento que debe contar el usuario de las mismas para poder utilizarlas, por lo que la siguiente tabla muestra el nivel de conocimiento requerido por el usuario administrador de las mismas.

Virtuoso	Axis	Servlets
• Instalar aplicación. • Configurar las aplicaciones cliente y servidor. • Modificar variables del entorno según el Sistema Operativo elegido. • Administrar en forma manual los archivos de configuración tanto para el cliente como para el Servidor.	• Instalar el servidor de aplicaciones. • Configuración manual del servidor de aplicaciones. • Configuración manual del entrono del Sistema Operativo. • Configuración manual de Axis. • Conocimientos para programar clases en Java con acceso a base de datos. • Necesario reinicializar aplicación para subir un controlador de conexión a base de datos extra (JDBC). • Conocimiento sobre la generación de documentos en XML.	• Instalar el servidor de aplicaciones • Configuración del Sistema operativo anfitrión. • Configuración de las variables de entorno para la plataforma Java. • Configuración de la aplicación en el servidor de aplicaciones. • Conocimientos de la arquitectura de Servlets. • Conocimientos para programar acceso a base de datos con Servlets. • Conocimientos para la generación de XML y su manipulación.

Figura 5.4 Requerimientos Técnicos.

Al revisar las especificaciones de cada aplicación y la tabla presentada en la figura 5.4, se determinó que el grado de conocimiento que un usuario debe tener es muy alto y no está solo orientado a la publicación de Servicios Web, si no también debe tener conocimientos de un lenguaje de programación (de acuerdo a la plataforma elegida), de configuración de sistema operativo, de programación orientada a acceso base de datos, entre otras.

Esta tesis fue creada tomando como una orientación encapsular todo este tipo de conocimientos y brindarle al usuario una aplicación en la que sólo es necesario:

- Ejecutar el archivo de instalación de la aplicación.
- Configurar la aplicación y no su entorno.

- No es necesario configurar el Sistema Operativo anfitrión.
- No es necesario saber algún Lenguaje de Programación para crear Servicios Web.

Al ser preferible que el usuario Administrador de base de datos utilice esta aplicación tendrá conocimientos de SQL, sin embargo no es un requisito para utilizar esta herramienta.

5.2.3 Tiempo de Respuesta

A pesar de que las pruebas realizadas en secciones anteriores presentan desventajas en la forma en requerimientos y necesidades, es necesario comparar también el tiempo de respuesta que brinda cada una de ellas.

Una vez que se han instalado y configurado las aplicaciones seleccionadas, se realizaron pruebas en dos partes, en donde la primera es el desarrollo de un Servicio Web y de los Servlets necesarios que accedieran a la base de datos y que generaran los resultados en XML; y la segunda es, una vez creados los elementos, se realizaron pruebas para saber el tiempo de respuesta ; sin embargo, debido a una de las restricciones de prueba, como se mencionó anteriormente, era que el consumidor no tuviera que programar y que pudiera acceder a través de una URL utilizando un explorador Web como IExplorer o NetScape.

Los resultados en base al desarrollo fueron los siguientes:

Virtuoso	Axis	Servlets
• Ofrece una interfaz gráfica importante, que sin embargo deben seguirse y configurarse diversos elementos para poder conseguir que se genere el resultado deseado.	• Configurar en el Servidor de Aplicaciones una aplicación para esta prueba. • Configurar la aplicación con las librerías necesarias. • Creación de una clase que sirvió como estructura para la publicación del Servicio Web. • Creación de la clase encargada de interactuar con la base de datos. • Creación de la clase que se encargará de convertir las consultas obtenidas en XML, en base a los metadatos de la base de datos.	• Instalar las librerías necesarias, tanto para base de datos como para XML en las variables de entorno del Sistema Operativo Anfitrión o en su caso en un directorio específico del Servidor de Aplicaciones (fue necesario reiniciar al Servidor). • Creación del Servlet controlador de las peticiones del usuario. • Creación de la clase encargada de interactuar con la base de datos. • Creación de la clase que se encargará de convertir las consultas obtenidas en XML, en base a los metadatos de la base de datos.

Figura 5.5 Desarrollo del Servicio.

Los resultados obtenidos muestran que Virtuoso fue de las aplicaciones que menos tiempo se necesitó invertir en el desarrollo de las pruebas, sin embargo Virtuoso en su versión de demostración no ofrece toda la potencialidad de esta aplicación.

Por otro lado al realizar un cambio de configuración o de la consulta realizada a la base de datos, generó la necesidad de modificar no sólo un segmento del código si no una parte importante de la misma.

El trabajo que se desarrolló, al estar orientado a base de datos, disminuyó el desarrollo necesario para publicar Servicios Web con acceso a base de datos y con el resultado en XML:

- Ya se cuenta con los módulos de acceso a base de datos.
- Cuenta con el módulo que convierte los resultados a XML.
- No se programó.
- La creación del Servicio Web se realizó a través de una ambiente gráfico.

Con esto el usuario de la aplicación se concentra en lo que desea publicar y no en la forma de hacerlo.

Por otro lado en cuento al tiempo de respuesta, se encontró que es variable debido a lo que debe realizar cada aplicación y a la forma en que la fuente de información es consultada, sin embargo se realizaron pruebas a cada aplicación las cuales se listan en la figura 5.6 y se obtuvieron los siguientes resultados:

Tipo de Consulta	Axis	Servlets	ASP	SW4BD
Consulta de una tabla con 100 registros	.351 seg	.300 seg	.400 seg	.370 seg
Inserción de 1 registro.	.310 seg	.250 seg	.300 seg	.35 seg
Modificación de un Registro.	.350 seg	.280 seg	.320 seg	.33 seg
Eliminación de 1 registro.	.160 seg	.150 seg	.180 seg	.20 seg
Consulta de una tabla con 10 registros.	.022 seg	.012 seg	.030 seg	.015 seg
Consulta de una vista con 50 registros.	.200 seg	.180 seg	.170 seg	.150 seg

Figura 5.6 Tiempo de Respuesta.

Los resultados mostrados en la figura 5.6 son muy similares, sin embargo todas las pruebas realizadas variaron mucho en base al tiempo transcurrido y en base a la estabilidad del servidor utilizado, por otra parte en las pruebas realizadas en tiempo de respuesta, el mismo aumentó cuando se realizaron peticiones en forma simultanea, esto es debido a la carga del servidor, sin embargo la parte importante de esta prueba fue verificar el compartimiento de este trabajo de tesis y observar que el tiempo de respuesta logrado varía de forma similar a otras opciones.

Cabe mencionar que para las aplicaciones de Axis, Servlets y ASP fue necesario crear una página anterior para poder manipular los parámetros enviados, por otro lado en Axis y Servlets se tuvieron que desarrollar métodos de validación de los parámetros enviados al proveedor. En cuanto a Virtuoso en esta tabla comparativa no aparece debido a que no ofrecía las características necesarias para tomar en cuenta la prueba, ya que contaba con varias restricciones al ser una aplicación con licencia. Se incluyó la generación de un documento XML utilizando ASP.

En el caso del trabajo desarrollado en esta tesis (SW4BD) se incluyo una página Web a través de la cual el consumidor de Servicios Web puede consultarlos en línea y puede manipular los parámetros que se tienen que enviar, por lo que no fue necesario crear páginas nuevas para consumirlos.

Aunque las aplicaciones para publicar Servicios Web ofrecen la posibilidad de consumirlos a través de un URL con parámetros, es complejo recordar los parámetros que se debe enviar; en cambio, cuando una aplicación ofrece la posibilidad de utilizar estos Servicios en línea y sin tener que construir URL parametrizadas, el servicio gana un aceptación, tal es el caso de Microsoft a través de ASPX, y de SW4BD.

5.3 Servidor de Servicios Web

Cuando ya se ha configurado al menos un Servicio Web, el servidor de Servicios Web se encarga de hacerlos disponibles para que los consumidores interactúen con ellos. De esta forma el servidor de Servicios Web, atiende las solicitudes enviadas por los consumidores y canaliza su atención en base al tipo de solicitud.

Para poder probar que el Servidor Web cumpliera con su cometido, se realizaron pruebas que indican como se realiza la comunicación entre el consumidor y el proveedor, además de revisar que el envío de recursos (WSDL principalmente) se realice de manera correcta; por otro lado se muestra la comunicación entre el cliente y el servidor a través de mensajes SOAP con o sin errores.

5.3.1 Configuración sin Autentificación

A través de la herramienta XML Spy [L-40] se realizó la petición de un documento de descripción del Servicio para ORACLE (archivo *OracleSW.wsdl*) según se muestra en la figura 5.7.

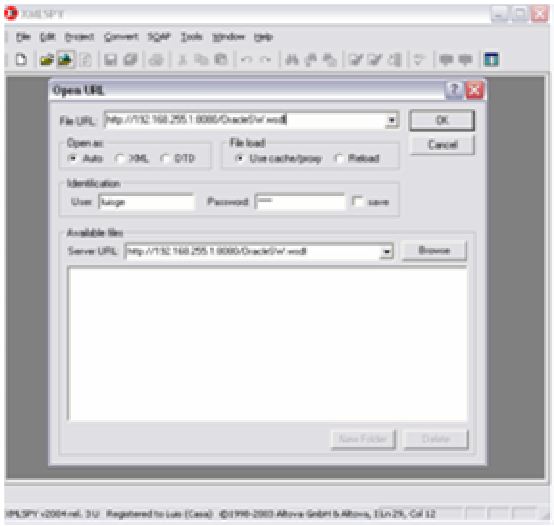


Figura 5.7 Petición de Documento WSDL.

La figura 5.8 muestra el resultado recibido en XML Spy. Como se observa el documento recibido es el archivo WSDL solicitado.

En base a este documento se realizó una prueba de ejecución de una operación del Servicio, *Listar_Alumnos*, que se encarga de consultar la tabla *Alumnos* de la base de datos ejemplo; el mensaje SOAP enviado es presentado en la figura 5.9.

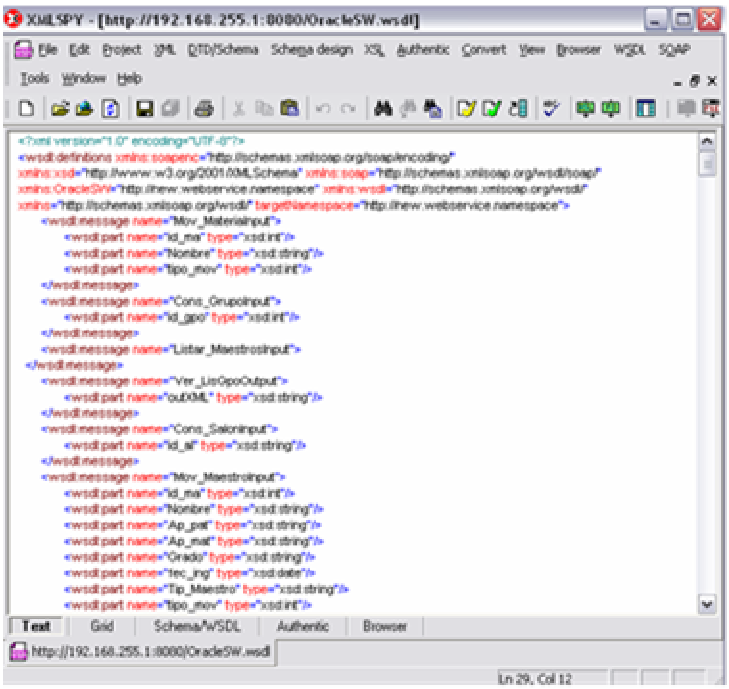


Figura 5.8 Documento WSDL Recibido.

Mensaje SOAP Enviado

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <m:Listar_Alumnos xmlns:m="urn:SW4BD:OracleSW" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.9 Mensaje SOAP Enviado.

El mensaje SOAP recibido es mostrado en la figura 5.10, donde el mensaje muestra una ejecución exitosa de la operación; el elemento *<SW4BD:Raiz>* contiene un documento XML con los datos de la tabla *Alumnos*.

Mensaje SOAP Recibido

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <soapenv:Body>
        <SW4BD:RESULT_SW4BD xmlns:SW4BD="urn:Listar_Alumnos">
            <SW4BD:Raiz>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&#xd;
&lt;Resultado_Operacion&gt;&lt;Res_Inst_0&gt;&lt;Registro&gt;&lt;ID_AL&gt;&lt;![CDATA[1]]&gt;&lt;/ID_AL&
gt;&lt;NOMBRE&gt;&lt;![CDATA[Franciso]]&gt;&lt;/NOMBRE&gt;&lt;APEPAT&gt;&lt;![CDATA[L&#243;pez]]&
gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Tovar]]&gt;&lt;/APEMAT&gt;&lt;FEC_NAC&gt;&lt;![CDATA[
1980-01-01 00:00:00.0]]&gt;&lt;/FEC_NAC&gt;&lt;FEC_ING&gt;&lt;![CDATA[2002-01-01
00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[M]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;Registr
o&gt;&lt;ID_AL&gt;&lt;![CDATA[2]]&gt;&lt;/ID_AL&gt;&lt;NOMBRE&gt;&lt;![CDATA[Beatriz]]&gt;&lt;/NOMBRE
&gt;&lt;APEPAT&gt;&lt;![CDATA[Gonz&#225;lez]]&gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Tello]]&
gt;&lt;/APEMAT&gt;&lt;FEC_NAC&gt;&lt;![CDATA[1982-02-22
```

```
00:00:00.0]]&gt;&lt;/FEC_NAC&gt;&lt;FEC_ING&gt;&lt;![CDATA[2000-08-01
00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[F]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;Registr
o&gt;&lt;ID_AL&gt;&lt;![CDATA[3]]&gt;&lt;/ID_AL&gt;&lt;NOMBRE&gt;&lt;![CDATA[Pamela]]&gt;&lt;/NOMBR
E&gt;&lt;APEPAT&gt;&lt;![CDATA[Hern&#225;ndez]]&gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Lucio
]]&gt;&lt;/APEMAT&gt;&lt;FEC_NAC&gt;&lt;![CDATA[1983-11-14
00:00:00.0]]&gt;&lt;/FEC_NAC&gt;&lt;FEC_ING&gt;&lt;![CDATA[1998-01-01
00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[F]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;Registr
o&gt;&lt;ID_AL&gt;&lt;![CDATA[4]]&gt;&lt;/ID_AL&gt;&lt;NOMBRE&gt;&lt;![CDATA[Gabriela]]&gt;&lt;/NOMBR
E&gt;&lt;APEPAT&gt;&lt;![CDATA[Esparza]]&gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Del

...

00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[F]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;Registr
o&gt;&lt;ID_AL&gt;&lt;![CDATA[13]]&gt;&lt;/ID_AL&gt;&lt;NOMBRE&gt;&lt;![CDATA[Arturo]]&gt;&lt;/NOMBR
E&gt;&lt;APEPAT&gt;&lt;![CDATA[Escalante]]&gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Lomel&#237
;]]&gt;&lt;/APEMAT&gt;&lt;FEC_NAC&gt;&lt;![CDATA[1985-10-16
00:00:00.0]]&gt;&lt;/FEC_NAC&gt;&lt;FEC_ING&gt;&lt;![CDATA[2000-01-01
00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[M]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;Registr
o&gt;&lt;ID_AL&gt;&lt;![CDATA[14]]&gt;&lt;/ID_AL&gt;&lt;NOMBRE&gt;&lt;![CDATA[Eircka]]&gt;&lt;/NOMBR
E&gt;&lt;APEPAT&gt;&lt;![CDATA[Cabrera]]&gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Landini]]&gt;&l
t;/APEMAT&gt;&lt;FEC_NAC&gt;&lt;![CDATA[1982-12-24
00:00:00.0]]&gt;&lt;/FEC_NAC&gt;&lt;FEC_ING&gt;&lt;![CDATA[2002-08-01
00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[F]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;Registr
o&gt;&lt;ID_AL&gt;&lt;![CDATA[15]]&gt;&lt;/ID_AL&gt;&lt;NOMBRE&gt;&lt;![CDATA[Adri&#225;n]]&gt;&lt;/NO
MBRE&gt;&lt;APEPAT&gt;&lt;![CDATA[Aguirre]]&gt;&lt;/APEPAT&gt;&lt;APEMAT&gt;&lt;![CDATA[Fonseca]]
&gt;&lt;/APEMAT&gt;&lt;FEC_NAC&gt;&lt;![CDATA[1982-09-01
00:00:00.0]]&gt;&lt;/FEC_NAC&gt;&lt;FEC_ING&gt;&lt;![CDATA[2000-01-01
00:00:00.0]]&gt;&lt;/FEC_ING&gt;&lt;SEXO&gt;&lt;![CDATA[M]]&gt;&lt;/SEXO&gt;&lt;/Registro&gt;&lt;/Res_In
st_0&gt;&lt;/Resultado_Operacion&gt;</SW4BD:Raiz>
            </SW4BD:RESULT_SW4BD>
        </soapenv:Body>
</soapenv:Envelope>
```


Figura 5.10 Mensaje SOAP Enviado como Resultado.

La figura 5.11 muestra el mensaje SOAP de la misma operación, pero con un error en el nombre de la operación y que deberá enviar como resultado un mensaje SOAP con la descripción del error ocurrido.

Mensaje SOAP Enviado

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <m:Lista_Alumnos xmlns:m="urn:SW4BD:OracleSW" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 5.11 Mensaje SOAP con Error en el Nombre de la Operación.

La figura 5.12 muestra el mensaje de error enviado por el Servidor de Servicios Web.

Mensaje SOAP Recibido

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <soapenv:Body>
        <soapenv:Fault>
            <faultcode>soapenv:Error en la ejecucion de la operacion solicitada</faultcode>
            <faultstring>La operacion LISTA_ALUMNOS no existe definida en el servicio
ORACLESW</faultstring>
            <faultactor>Atendiendo Servicio</faultactor>
            <detail/>
        </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

Figura 5.12 Mensaje SOAP con Error Recibido.

5.3.2 Mensajes recibidos en base a la Configuración

Utilizando las herramientas XML Spy e Internet Explorer se realizaron diversas peticiones para poder revisar el resultado que arrojaba el Servidor de Servicios Web.

En ambas aplicaciones se realizó la solicitud del mismo documento WSDL, con la diferencia que la configuración actual del servidor indicaba que debería utilizarse la autentificación para acceder al recurso.

La figura 5.13 muestra como Internet Explorer también solicita al usuario se autentifique para poder acceder al recurso.

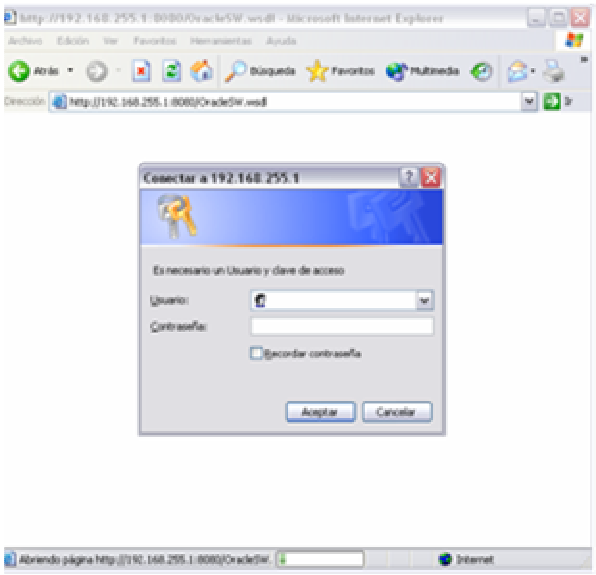


Figura 5.13 Autentificación en Internet Explorer.

La figura 5.8 muestra el recurso solicitado y que fue obtenido mediante la autentificación correcta en XML Spy; la figura 5.14, muestra el mensaje enviado por el Servidor una vez que no se aceptó lo autentificación por el usuario.

Figura 5.14 Acceso Denegado.

En caso de que el recurso solicitado no sea válido o no esté disponible, el consumidor de Servicios Web, recibirá el mensaje mostrado en la figura 5.15.

Figura 5.15 Recurso no Disponible.

5.4 Aplicaciones de Prueba

Una de las partes principales de la aplicación es el de atender las solicitudes enviadas por los consumidores y de dar servicio no sólo a un consumidor, es por ello que en esta sección se presentan dos aplicaciones que fueron construidas para probar esta aplicación.

La primera fue desarrollada para probar la forma en que son atendidas las solicitudes y la segunda es una pequeña aplicación Web realizada con JavaScript que permitió probar los mensajes SOAP recibidos con datos en XML y poder trabajar con ellos.

5.4.1 Aplicación para verificar Solicitudes

Esta aplicación se encarga de enviar diferentes solicitudes al servidor de manera concurrente. En ella se muestran 9 pantallas que indican que solicitud se está enviando y cual es el resultado recibido; la figura 5.16 muestra la pantalla principal de la aplicación.

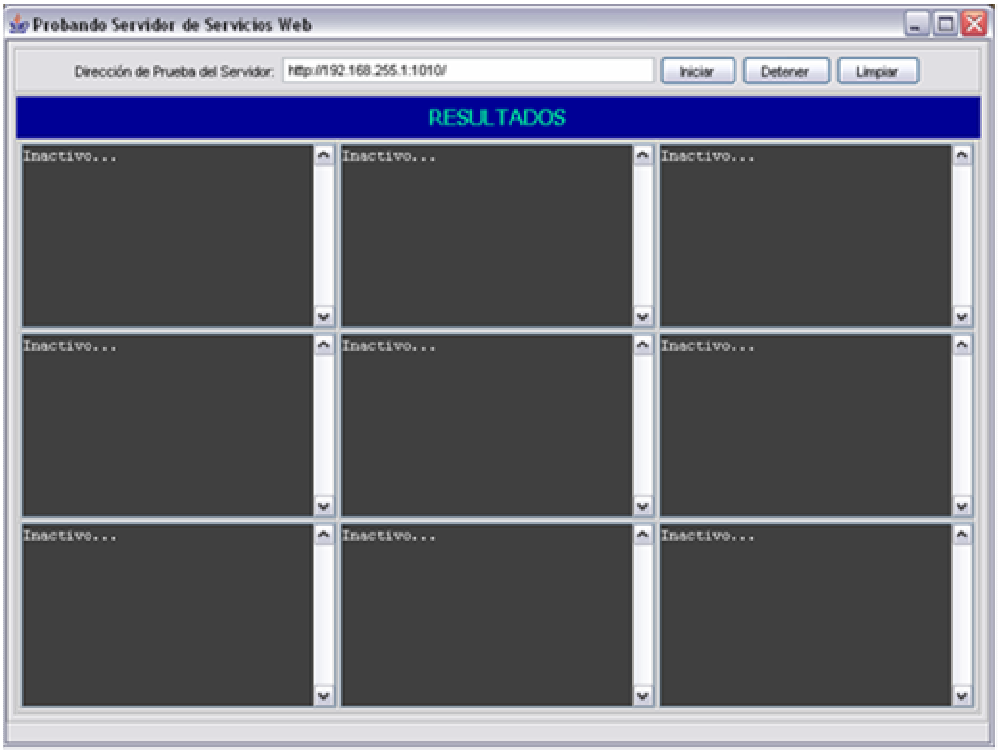


Figura 5.16 Pantalla Principal para Probar Solicitudes.

Las solicitudes enviadas corresponden a las solicitudes de recursos válidos, así como no válidos en forma aleatoria, además de enviar mensajes SOAP de ciertas operaciones para su atención y muestra de los resultados.

5.4.2 Página Web con Servicios Web y JavaScript

El objetivo principal de ésta aplicación fue la de crear un entorno de pruebas donde se enviarán peticiones de ejecución de operaciones de un Servicio Web en específico (ORACLE) y que a través de su funcionamiento verificar que se recibe la información en XML y puede ser transformada para visualizarla en un navegador de HTML.

Para realizar ésta aplicación se utilizó el envío directo de mensajes SOAP y para la interpretación de los datos en XML recibidos a XSL; la figura 5.17 muestra la pantalla principal de ésta aplicación.

Figura 5.17 Pantalla Principal de la Aplicación Consumidora.

En esta aplicación cliente se muestra como a través de consultas a Servicios Web que arrojan los resultados en XML se puede lograr que una aplicación pueda consultar bases de datos heterogéneas e interactuar con ellas.

El usuario de esta aplicación puede seleccionar la fuente de información en la cual desea estar trabajando, en este caso MySQL, PostgreSQL u Oracle, cabe mencionar que no importa la forma en que este creada la base de datos, la parte importante es al momento de construir el cuerpo de las operaciones de los Servicios Web, pues el resultado en XML

es conformado según la configuración de los mismos, es por ello que puede crearse un documento XML similar para cada fuente de información.

En esta aplicación se consumen Servicios Web que se encargan de consultar y manipular la base de datos, esto es existen servicios que arrojarán consultas y servicios que alterarán los datos de la misma (altas, bajas y modificaciones).

La dinámica de la aplicación se realizó mediante JavaScript y el envío de mensajes SOAP utilizando a MS SOAP Toolkit y también enviando de manera directa los mensajes SOAP. Utilizar JavaScript para realizar una aplicación cliente muestra que no es necesario tener un Servidor de aplicaciones que contengan a esta aplicación, cualquier computadora con una comunicación al Servidor de Servicios Web podrá hacer uso de esta aplicación.

5.5 Resumen

En este capítulo se presentaron algunas de las pruebas realizadas a la aplicación presentada en este trabajo, tomando en cuenta que las pruebas se orientaron a verificar aquellas partes que hacían a esta herramienta genérica como las conexiones JDBC, la creación de los archivos de configuración XML, el envío y recepción de solicitudes en la parte del servidor. Por otro lado se realizaron pruebas con dos aplicaciones cliente para consumir Servicios Web y poder mostrar los resultados.

Capítulo VI
Conclusiones y Trabajos Futuros

6.1 Conclusiones

En el contexto de las bases de datos, uno de los principales retos ha sido lograr su interoperabilidad. Con el surgimiento de Internet, lograr la interoperabilidad ha sido más complejo debido a la variedad de formatos existentes para la publicación de los mismos.

El realizar este trabajo permitió conocer las diferencias importantes existentes en los Sistemas Administradores de Base de Datos (SABD) que dificultan lograr la interoperabilidad de los mismos, incluso utilizando estándares de conectividad como son JDBC y ODBC.

Es evidente que los Servicios Web han tenido un impulso importante en los últimos meses y al contar con la tecnología suficiente para convertir aplicaciones corporativas a Servicios Web se logra alcanzar un alto nivel de interoperabilidad. Por otro lado, existen diversos entornos de desarrollo para generar Servicios Web, aunque la mayoría están basados en los estándares que definen a los Servicios Web (SOAP y WSDL, principalmente), con ello se observa que no es importante en que plataforma desarrollemos el Servicios Web y tampoco en cual plataforma consumamos dicho Servicio, ya que mientras se utilicen estos estándares se asegurará la interoperabilidad entre estos.

Esta tecnología es muy importante en el campo de la tecnología de información y ha crecido constantemente y a pasos rápidos en los últimos meses; sin embargo se espera que su crecimiento y aceptación sean mayores al contar con niveles superiores de seguridad e integración, como por ejemplo el desarrollo de una Arquitectura Orientada a Servicios (SOA por sus siglas en inglés Service Oriented Architecture).

A pesar que los Servicios Web están teniendo una aceptación importante, las herramientas para generarlos presentan cierto nivel de complejidad a los usuarios de las mismas, además se debe contar principalmente con conocimientos de programación, por lo que este trabajo de tesis podrá tener una aceptación mayor al eliminar la necesidad de conocer un lenguaje de programación.

Actualmente uno de los trabajos en el que mayor tiempo se invierte ha sido construir aplicaciones que se encarguen de unificar la información consultada en diferentes fuentes de información (comúnmente con diferentes niveles de heterogeneidad), por ello este trabajo se orienta a simplificar el desarrollo de estas herramientas y brindar una plataforma que fortalezca la interoperabilidad de las mismas.

El desarrollo de esta tesis permite que exista interoperabilidad en varios niveles entre plataformas heterogéneas. En la parte de comunicación se utiliza Servicios Web y en el envío de resultados el uso de XML.

Actualmente la fusión de las empresas y la globalización reinciden en la necesidad de intercambiar información entre diversas compañías e incluso entre sistemas de una misma empresa, provocando que esta herramienta encuentre un nicho importante para su uso.

En la conceptualización y desarrollo de este trabajo se observó que las empresas líderes en el desarrollo de tecnologías para la administración de información (principalmente SABD) han orientado sus esfuerzos en ofrecer características que son similares a las que aquí se cubren (Servicios Web y XML).

Al ser XML un estándar para el intercambio de información y el cual ha sido aceptado en forma importante por los desarrolladores de la tecnología de información, es importante contar con herramientas que permitan manipularlo y que transformen los resultados de las consultas de los SABD a XML. Además existen una gran variedad de tecnologías relacionadas con XML y que han surgido principalmente para facilitar la manipulación de este formato, como XSL, XPATH y XQuery. En este proyecto se utilizó a XPATH para poder localizar uno o varios elementos dentro de un documento XML, facilitando la administración del mismo.

XSL permite al consumidor de Servicios Web generados en este proyecto, manipular el documento XML arrojado como resultado de la ejecución de alguna operación de estos servicios, con ello se podrá presentar la misma información en el formato deseado e incluso facilitar a los consumidores el análisis de la información publicada para posteriormente crear sus propios documentos de información de resultados.

La creación de esta aplicación permitió conocer las ventajas y debilidades de las tecnologías que actualmente han tenido un auge mayor y conocer a fondo los trabajos que se están realizando para lograr interoperabilidad entre sistemas heterogéneos.

Por lo anterior la creación de herramientas genéricas (en este caso orientada a base de datos) es compleja y se deben tomar decisiones sobre los alcances de la misma, debido a la falta de interoperabilidad entre los estándares utilizados en la parte de conectividad a las bases de datos, ya que la mayoría de los fabricantes de SABD y que emiten estos estándares agregan funcionalidad (adicional a la especificación) para acceder a su SABD.

6.2 Resultados

Con el desarrollo de esta aplicación se lograron alcanzar los siguientes objetivos:

- Brindar una aplicación que facilite la creación de Servicios Web con acceso a base de datos, utilizando los estándares referentes a ésta tecnología para el empaquetado, procesamiento y envío de resultados.
- Ofrecer una herramienta moderna, simple y novedosa, que permita publicar la información de los SABD a través de Servicios Web sin importar la plataforma de desarrollo del consumidor de los mismos y orientada a un propósito.
- Se logró implementar las partes esenciales de los Servicios Web como son el envío y recepción de mensajes SOAP y la generación de los documentos, que describen los Servicios Web (WSDL).
- Crear un Servidor de Servicios Web ligero orientado solo a la atención de consultas a bases de datos, con lo cual el usuario de esta herramienta no necesita contar con otras tecnologías para poder utilizarlas.

6.3 Trabajos Futuros

En esta sección se presentan algunas extensiones probables que podrían realizarse a este trabajo para futuros proyectos:

- Crear el cuerpo de cada operación, donde cada instrucción SQL este relacionada con el mismo o diferente conexión JDBC; esto es, que a través de una sola operación de un Servicio Web puedan consultarse diferentes SABD, que sea transparente para el consumidor.
- Permitir que un Servicio Web pueda conectarse a diferentes conexiones JDBC para poder atender de manera oportuna las peticiones de los consumidores.
- Adecuar la aplicación a las nuevas especificaciones sobre Servicios Web, principalmente aquellas que tienen que ver con aspectos de Seguridad
- Crear en forma automática para cada conexión, Servicios Web encargados a publicar las características de la base de datos, obtener los elementos que la componen, las características de cada uno de ellos, entre otras funciones.
- Permitir crear operaciones que obtengan como parámetros documentos XML tanto para la inserción de información como para la manipulación, esto es que a través de un documento XQuery recibido como parámetro se realicen operaciones en el SABD y se comunique el resultado en XML.

- Crear una librería que permite comunicar a esta aplicación con los diferentes Servidores Web existentes en el mercado (IIS, Apache, Tomcat, WebSphere, entre otros).
- Interactuar con los diversos servidores UDDI existentes para poder publicar los Servicios Web creados en esta aplicación.
- Actualizar la conexión JDBC utilizada a la especificación 3.0 para permitir el uso de *DataSource* en base de realizar la conexión como actualmente se esta llevando a cabo y así poder contar con un pool de conexiones mayor a la presentada actualmente.

6.4 Comentarios Finales

La realización de este trabajo llevó al aprendizaje e investigación de las tecnologías como XML, XSL, Servicios Web, Java, SOAP, WSDL, XML-RPC, entre otras que estaban relacionadas con el intercambio de información y la ejecución remota de procesos.

Además de conocer diferentes plataformas de desarrollo de Servicios Web para poder interactuar con esta herramienta y conocer el grado de interoperabilidad de la misma. Por otro lado la experiencia ganada en el desarrollo de la aplicación, al conocer las diferentes implementaciones de Servicios Web que existen y las necesidades de las mismas para poderlas utilizar, como son Axis de Jakarta, Apache SOAP, el toolkit de Java para Servicios Web, .Net de Microsoft y las necesidades de los mismos para poder utilizarlos para crear Servicios Web que acceden a base de datos.

El aprendizaje obtenido del manejo de diferentes SABD que presentan características interesantes y que permitieron poder conocer que tecnologías ofrecen, la forma en que deben ser configurados para poder ofrecerlas, sus necesidades externas para configurarlos.

Como comentario final, la utilización de estas tecnologías ha permitido conocer las novedades y las tendencias en el contexto de la tecnología de información y ha ayudado al desarrollo en el aspecto profesional, pues en la experiencia obtenida en el aspecto laborar, observado que estas tecnologías han sido adoptadas y usadas de manera importante en los proyectos que se ha participado; es por ello que se ha podido constatar y entender mejor la aplicación de Servicios Web, y saber que tendrán un importante crecimiento en los meses venideros. Es por ello que es satisfactorio presentar un trabajo orientado a la utilización de estas tecnologías y alineado a las tendencias de desarrollo actuales.

ANEXO A
JDBC

En este anexo se presenta con más detalle la tecnología utilizada para interactuar con los SABD que serán utilizados para publicar sus datos como Servicios Web.

Forma en que JDBC Accede a las Bases de Datos

Para poder acceder a los datos contenidos en la fuente, es necesario realizar dos tareas elementales y que son las que nos permitirá posteriormente podernos conectar y manipular los datos de la fuente elegida:

Registrar el Controlador: permite que el administrador de controladores mantenga una referencia a todas las características que proporcione el controlador JDBC. Para registrar el controlador es necesario cargarlo a través de su clase principal; la figura A.1 muestra un ejemplo para conectarse a bases de datos contenidas en diferentes SABD:

Código General

```
…
String jdbcDriver = "org.gjt.mm.mysql.Driver";
String dbURL ="jdbc:mysql://localhost/phonebook";
Connection dbConn = null;
try
{
  Class.forName(jdbcDriver).newInstance();
  …
}
catach(SQLException e)
{
   …
}
```

Figura A.1 Instalando Controlador JDBC.

Una vez realizado este paso el controlador queda registrado y se puede trabajar con las características que ofrece.

Crear una conexión con la base de datos: La conexión a la base de datos será la encargada de encaminar las solicitudes a la base de datos y de recuperar los resultados y/o errores generados por las mismas.

Sistema Administrador de Base de Datos	Valor de la variable jdbcDriver
SQL Server	com.microsoft.jdbc.sqlserver.SQLServerDriver
ORACLE	oracle.jdbc.driver.OracleDriver
PostgreSQL	org.postgresql.Driver
MySQL	org.gjt.mm.mysql.Driver

Figura A.2 Controladores utilizados para los SABD.

Para realizar la conexión es necesario contar básicamente con tres datos:

- **URL** el cual le indica al controlador dónde debe buscar la fuente de datos.

jdbc:SUBPROTOCOL:SUBNAME

  - **SUBPROTOCOL:** indica el nombre válido de un controlador JDBC u otra solución de conectividad a base de datos.
  - **SUBNAME:** indica un nombre lógico o un seudonombre de la base de datos a la que se conectará.

La cadena de conexión utilizada dependerá de la fuente de datos deseada y del controlador utilizado. La figura A.3 muestra la cadena de conexión para los SABD utilizados en este proyecto.

Sistema Administrador de Base de Datos	Cadena de conexión
SQL Server	Jdbc:microsoft:sqlserver://tlaloc.cic.ipn.mx:3314
ORACLE	Jdbc:oracle:thin:@tlaloc.cic.ipn.mx:1521:Tesis
PostgreSQL	Jdbc:postgresql:// tlaloc.cic.ipn.mx /Tesis
MySQL	Jdbc:mysql:// tlaloc.cic.ipn.mx /Tesis

Figura A.3 Cadena de Conexión JDBC.

Además de la cadena de conexión, será necesario proporcionar un nombre de usuario y una clave de acceso para poderse autentificar en la base de datos.

Una vez que se logró una conexión a la base de datos, se podrá administrar dicha fuente utilizando las interfaces, clases y métodos ofrecidos por el controlador, tomando en cuenta que no todas las que se describen en la especificación JDBC estarán disponibles.

Principales Interfaces y Clases de JDBC

La figura A.4 muestra un diagrama de cómo se relacionan estas interfaces y clases; algunas no se presentan en el diagrama pero se describe porque son de utilidad para este proyecto.

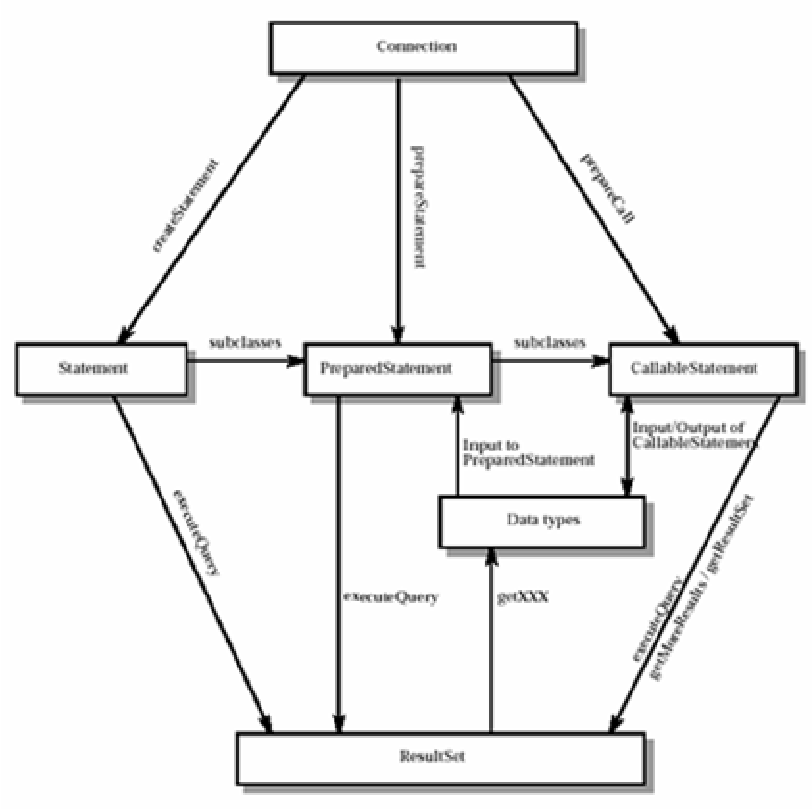


Figura A.4 Relación de las Clases para JDBC.

Connection

Un objeto Connection representa una conexión a una base de datos. Se encarga de ser el medio por el cual las sentencias SQL serán enviadas a la fuente de datos, así como por donde la aplicación recibirá los resultados y/o errores producidos por las sentencias ejecutadas.

La forma de establecer una conexión es a través de la clase administradora de controladores DriverManager, la cual cuenta con diferentes tipos de métodos para generar un objeto Connection.

Siguiendo con la base de datos ejemplo, para obtener una conexión a la base de datos Tesis, se utiliza el fragmento de código mostrado en la figura A.5.

Conexión

```
…
String jdbcDriver = "org.gjt.mm.mysql.Driver";
String dbURL ="jdbc:mysql://localhost/phonebook";
Connection dbConn = null;
try
{
  Class.forName(jdbcDriver) .newInstance() ;
  Connectioncon=DriverManager.getConnection("jdbc:mysql://
  tlaloc.cic.ipn.mx /Tesis ",usuario, claveacceso);
  …
}
catach(SQLException e)
{
   …
}
```

Figura A.5 Conectando a la base de datos.

En este ejemplo obtenemos un objeto Connection, para posteriormente utilizar con sus métodos que ofrece para trabajar con la base de datos. Las variables usuario y claveacceso deben cambiarse con los valores correctos. La figura A.6 lista algunos de los métodos más utilizados para esta interfaz:

Método	Descripción
close()	Cierra la conexión con la base de datos
commit()	Envía una instrucción commit, para que se guarden los cambios realizados a la base de datos.
createStatement()	Crea un objeto *Statement*, para poder realizar sentencias SQL.
getAutoCommit()	Obtiene el estado actual del commit, si es verdadero, todas las operaciones que se realicen a la base de datos, se aplicaran de forma inmediata.
getCatalog()	Indica el catálogo o base de datos a la que se está conectado actualmente, dependerá si el SABD soporta estas características.
getMetaData()	Obtiene la información de la fuente de datos a donde se conecta, consulta la descripción de la interfaz

	DatabaseMetaData.
getTransactionIsolation()	Obtiene el grado de aislamiento de las transacciones.
prepareStatement()	Crea una sentencia SQL, pero con la opción de poderse utilizar tantas veces como se requiera, con ella se pueden realizar consultas con argumentos; son ejecutadas con mayor velocidad que los *Statement*.
prepareCall()	Genera un objeto para poder realizar llamadas a procedimientos almacenados.
rollback()	Si el commit dependerá de la aplicación y no del SABD, este método desecha las operaciones realizadas desde el último commit.
setAutoCommit()	Cambio el estado del *Autocommit*.
setCatalog()	Cambia el catálogo con que se está trabajando en la base de datos.

Figura A.6 Métodos de la Interfaz Connection.

Es importante aclarar que sólo se presenta el nombre y la descripción de cada método, para saber que tipo de parámetros necesita y su utilización, consultar la especificación de JDBC localizada en [L-39].

PreparedStament

Un objeto *PreparedStatement* representa una sentencia SQL compilada, por lo que su velocidad de ejecución es mayor que un objeto *Statement*.

Las diferencias entre estos dos objetos son:

- Los objetos *PreparedStatement* contienen sentencias SQL que ya han sido compiladas, lo que da su nombre de sentencias preparadas.
- La sentencia SQL que contiene un *PreparedStatement* puede tener o no argumentos de entrada, por lo que permite ejecutar la misma sentencia con diferentes argumentos utilizando los métodos *setXXX*.

La figura A.7 presenta un ejemplo para obtener el nombre de un empleado utilizando un objeto *PreparedStatement*.

PreparedStatement

```
…
PreparedStatement consulta = Conn.PreparedStatement("select
nombre from empleado where id=?");
Consulta.setInt(109);
ResultSet rs = consulta.executeQuery();
…
System.out.println(rs.getString("NOMBRE"));
...
Consulta.setInt(150);
ResultSet rs = consulta.executeQuery();
…
System.out.println(rs.getString("NOMBRE"));
...
```

Figura A.7 Objeto PrepardStatement.

Los métodos principales de esta interfaz son listados en la figura A.8.

Método	Descripción
clearParameters()	Limpia los parámetros actuales de la sentencia SQL.
executeQuery()	Ejecuta la sentencia preparada como sentencia de consulta (regresa un objeto *ResultSet*).
executeUpdate()	Ejecuta una sentencia de actualización (regresa un valor entero indicando los registros afectados).
getMetaData()	Obtiene la información de los metadatos del *ResultSet* que será retornado después de la ejecución de la sentencia.
getParameterMetaData()	Obtiene la información de los metadatos de los parámetros enviados a la sentencia.

Figura A.8 Métodos de la Interfaz *PreparedStatement*.

CallableStatement

Un objeto *CallableStatement* es la forma en la que se pueden realizar llamadas a procedimientos almacenados. La sentencia de llamada para procedimientos almacenados está escrita con sintaxis de escape, la cual se puede realizar con parámetros donde se obtienen los resultados o sin estos parámetros.

Un ejemplo para utilizar esta sentencia se muestra en la figura A.9.

CallableStatement

```
…
CallableStatement proc = conn.prepareCall("{call
getTestData(?,?)}");
…
```

Figura A.9 Creando un Objeto CallableStatement.

Una vez creado el objeto *CallableStatement*, es posible definir los parámetros que complementarán a la sentencia que será enviada al SABD, estos se definen mediante los métodos *setXXX*, los cuales se determinan en base al tipo de parámetro deseado.

Statement

Es la interfaz más básica de las tres que pueden enviar sentencias SQL y recuperar resultados. Al ser una interfaz básica, se utiliza para enviar sentencias SQL simples sin parámetros.

Brinda métodos básicos para ejecutar sentencias y recuperar resultados. Para crear un objeto *Statement* se puede realizar mediante el código de la figura A.10:

Statement

```
…
Statement consulta = Conn.createStatement();
ResultSet rs = consulta.executeQuery("select nombre from
empleado where id=109");
…
System.out.println(rs.getString("NOMBRE"));
…
```

Figura A.10 Creando un Objeto Statement.

Los métodos que brinda esta interfaz son listados en la figura A.11.

Método	Descripción
addBatch()	Agrega una consulta SQL a un conjunto de sentencias que serán ejecutadas en lotes.
close()	Cierra un objeto *Statement*, si este objeto tiene objetos *ResultSet* relacionados también serán cerrados.
executeBatch()	Ejecuta un grupo de sentencias establecidas con

	addBatch().
executeQuery()	Ejecuta una sentencia de consulta SQL y regresa un objeto *ResultSet* con los datos obtenidos.
executeUpdate()	Ejecuta una sentencia de actualización SQL y regresa un valor entero indicando el número de registros afectados.

Figura A.11 Métodos de la Interfaz Statement.

ResultSet

Un objeto ResultSet es una tabla que contiene los resultados después de ejecutar una sentencia SQL de consulta. Los datos que son almacenados en un objeto *ResultSet* pueden ser recuperados a través de los métodos de lectura *getXXX*, donde *XXX* representa el tipo de dato que corresponde al que se desea recuperar. Un objeto *ResultSet* puede ser generado a través de diversos métodos de las interfaces presentadas en este documento, la forma más utilizada de crear un objeto *ResultSet* es a través de la ejecución de una sentencia SQL de consulta.

Un ejemplo de cómo recorrer los resultados obtenidos a través de la ejecución de una consulta SQL se muestra en el código de la figura A.12.

ResultSet

```
…
Statement consulta = Conn.createStatement();
ResultSet rs = consulta.executeQuery("select nombre from
empleado where id>=109");
…
while (rs.next())
System.out.println(rs.getString("NOMBRE"));
…
```

Figura A.12 Obteniendo datos de un ResultSet.

Método	Descripción
absolute()	Posiciona en un registro específico del objeto ResultSet.
close()	Cierra un objeto ResultSet.
findColumn()	Obtiene de los metadatos el número de columna indicado con su nombre.
first()	Posiciona en el primer registro objeto *ResultSet*
getMetaData()	Obtiene los metadatos de un objeto *ResultSet*, regresa un objeto *ResultSet*.
next()	Posiciona al objeto *ResultSet* en el registro siguiente.
previous()	Posiciona al objeto *ResultSet* en el registro anterior.

Figura A.13 Métodos de la Interfaz ResultSet.

DatabaseMetaData

Una de las interfaces más importantes de la especificación de JDBC es *DatabaseMetaData*, ya que esta interfaz nos indica la conformación de la fuente de datos con la cual se está trabajando. Esta interfaz permite recuperar información sobre los catálogos existentes, las bases de datos existentes, las características que brinda la fuente de datos a la que se conecta, la estructura de cada una de las tablas y la descripción de los objetos contenidos en la base de datos actual, entre otras características.

El código de la figura A.14 es un ejemplo de cómo desplegar las tablas de la base de datos actual y cada una de las columnas que conforman a las tablas:

DatabaseMetaData

```
...
DatabaseMetadata dbmd = con.getMetaData();
String[] s = {"TABLE","VIEW"};
ResultSet tablas = dbmd.getTables("Tesis","%","%",s);
while (tablas.next()){
  System.out.println("TABLA:"+tablas.getString("TABLE_NAME"));
   ResultSet campos=dbmd.getColumns(null,null,"Empleados","%")
    while (campos.next())    {
System.out.println("Columna:"+campos.getString("COLUMN_NAME"));
    }
}
...
```

Figura A.14 Ejemplo para obtener los Metadatos.

Método	Descripción
allProceduresAreCallable()	Determina si todos los Procedimientos Almacenados obtenidos con el método getProcedures() pueden ejecutarse.
getCatalogs()	Obtiene un objeto *ResultSet* con las bases de datos presentes en la fuente de datos.
getColumns()	Obtiene un objeto *ResultSet* con la información de cada columna de una Tabla.
getExportedKeys()	Obtiene un objeto *ResultSet* con la información de cada llave foránea.
getImportedKeys()	Obtiene un objeto *ResultSet* con la información de cada llave primaria referenciada en la tabla.
getPrimaryKeys()	Obtiene un objeto *ResultSet* con la información de cada llave primaria de la tabla indicada.
getProcedures()	Obtiene un objeto *ResultSet* con información de los procedimientos almacenados en la fuente de datos.
getTables()	Obtiene un objeto *ResultSet* con información referente a las tablas presentes en la base de datos indicada.
supportsStoredProcedures()	Indica si la base de datos a la que se esta conectando soporta procedimientos almacenados.

Figura A.15 Métodos de la Interfaz DatabaseMetadata.

Tipos de Datos

Existe una diversidad de tipos de datos en las bases de datos a las cuales se conectan los controladotes JDBC, por lo que es necesario saber que tipos de datos corresponden en ambas tecnologías. La siguiente es una lista de los tipos de datos Java a los que se puede homologar los tipos de datos que ofrezca la base de datos seleccionada, además de los métodos utilizados para obtener los datos o para fijar los datos en una sentencia SQL.

Tipo de dato SQL	Método Java para Leer	Método Java para Escribir
Inyint	getByte()	setByte()
Smallint	getInt()	setInt()
mediumint	getInt()	setInt()

int ó integer	getInt()	setInt()
bigint long	getLong()	setLong()
real	getFloat()	setFloat()
float	getFloat()	setFloat()
double	getDouble()	setDouble()
decimal	getBigDecimal()	setBigDecimal
numeric	getBigDecimal()	setBigDecimal
number	getBigDecimal()	setBigDecimal
bit	getBoolean()	setBoolean()
char or character	getString()	setString()
varchar	getString()	setString()
longvarchar	getAsciiStream()	setAsciiStream()
clob	getClob()	setClob()
text	getAsciiStream()	setAsciiStream()
binary	getBytes()	setByte(), setBytes()
varbinary	getBytes()	setByte(), setBytes()
longvarbinary	GetBinaryStream()	setBinaryStream()
blob	getBlob()	setBlob()
date	getDate()	setDate()
datetime	getDate()	setDate()
time	getTime()	setTime()
timestamp	getTimestamp()	setTimestamp()

Figura A.16 Tipo de datos, métodos *getXXX* y *setXXX*.

Los tipos de datos pueden diferir de acuerdo al SABD utilizado, por lo que se recomienda revisar la documentación técnica para el SABD utilizado.

JDBC contempla los tipos de datos estándares y que se presentan en la mayoría de los SABD, quizás con normes diferentes en los tipos de datos pero que se utilizan de la misma manera en todos; sin embargo existen SABD que permiten la definición de nuevos tipos tal como lo hace PostgreSQL, donde el usuario puede definir tipos de datos y almacenarlos en una tabla.

ANEXO B
WSDL

WSDL es un documento XML que describe la interfaz, semántica y administración de llamada al Servicio Web. Las operaciones y los mensajes se describen de manera abstracta y después se enlazan a un protocolo de red y aun formato de mensaje concreto.

En este anexo se presentan los principales elementos que debe contener un documento WSDL para describir los Servicios Web en general, los elementos aquí presentados no son todos los que puede contener una descripción de servicio, para saber la especificación completa del WSDL consulte la especificación mostrada en [L-11].

Estructura de Documento WSDL

Los documentos WSDL definen los servicios como colecciones de puntos de entrada en la red o puertos. En WSDL, la definición abstracta de puntos de entrada y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas:

- **Mensajes:** que son descripciones abstractas de los datos que se están intercambiando.
- **Tipos de puertos:** que son colecciones abstractas de operaciones.

Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types:** contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos.
- **Message:** definición abstracta y escrita de los datos que se están comunicando.
- **Operation:** descripción abstracta de una acción admitida por el servicio.
- **Port Type:** conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding:** especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port:** punto final único que se define como la combinación de un enlace y una dirección de red.

- **Service:** colección de puntos finales relacionados.

En ejemplo de la uso de estos elementos es el mostrado en la figura B1. En este trabajo algunos elementos no son utilizados para la generación del documento WSDL, por lo que no son explicados en este anexo.

Ejemplo de Documento WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ServicioMySQL="http://new.webservice.namespace" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://new.webservice.namespace">
    <wsdl:message name="Operacion1Input">
 </wsdl:message>
    <wsdl:message name="Operacion1Output">
        <wsdl:part name="outXML" type="xsd:string"/>
    </wsdl:message>
    <wsdl:portType name="ServicioMySQL_port">
        <wsdl:operation name="Operacion1">
            <wsdl:input name="Operacion1Input" message="ServicioMySQL:Operacion1Input"/>
            <wsdl:output name="Operacion1Output" message="ServicioMySQL:Operacion1Output"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="ServicioMySQL_binding" type="ServicioMySQL:ServicioMySQL_port">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="Operacion1">
            <soap:operation soapAction="http://luis_movil:1010/WSDataBase/ServicioMySQL?Operacion1"/>
            <wsdl:input>
                <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SW4BD:ServicioMySQL"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SW4BD:ServicioMySQL"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ServicioMySQL_service">
        <wsdl:port name="ServicioMySQL_Port" binding="ServicioMySQL:ServicioMySQL_binding">
            <soap:address location="http://luis_movil:1010/WSDataBase/ServicioMySQL"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Figura B.1 Documento WSDL de ejemplo.

Es importante observar que WSDL no introduce un nuevo lenguaje de definición de tipos. WSDL reconoce la necesidad de disponer de diferentes sistemas de tipos para describir los formatos de mensaje y admite como sistema de tipos la especificación de los esquemas XML. Sin embargo, puesto que no es razonable esperar una única gramática del sistema de tipos que se utilice para describir todos los formatos de mensajes presentes y futuros, WSDL permite el uso de otros lenguajes de definición de tipos mediante la extensibilidad.

Mensajes

Los mensajes se componen de una o más partes lógicas. Cada parte está asociada a un tipo procedente de algún sistema de tipos que utiliza un atributo de escritura de mensajes. El conjunto de atributos de escritura de mensajes es extensible. WSDL define varios atributos de este tipo para su uso con XSD (XMLSchema Definition, Definición de Esquema XML):

- **Element:** Hace referencia a un elemento XSD que utiliza un QName.
- **Type:** Hace referencia a un tipo *simpleType* o *complexType* de XSD que utiliza un QName.

Se pueden definir otros atributos de escritura de mensajes siempre que utilicen un espacio de nombres distinto al de WSDL. Los elementos de extensibilidad del enlace también pueden utilizar atributos de escritura de mensajes.

La sintaxis para definir un mensaje se muestra en la figura B.2.

Ejemplo de Uso del Elemento Mensajes

```
<wsdl:message name="Operacion1Input">
 </wsdl:message>
 <wsdl:message name="Operacion1Output">
    <wsdl:part name="outXML" type="xsd:string"/>
 </wsdl:message>
```

Figura B.2 Elemento Mensajes.

El atributo *name* de *message* proporciona un nombre único entre todos los mensajes definidos en el documento WSDL adjunto.

El atributo *name* de *part* proporciona un nombre único entre todas las partes del mensaje adjunto.

Partes del mensaje

Las partes de un mensaje son un mecanismo flexible para describir su contenido abstracto y lógico. Un enlace puede hacer referencia al nombre de una parte, con el fin de precisar información específica de enlaces sobre dicha parte. Por ejemplo, si se define un mensaje para su uso con RPC, una parte puede representar un parámetro del mensaje. Sin embargo, los enlaces deben inspeccionarse para determinar el significado real de la parte.

En estos mensajes se pueden definir partes con tipos de datos complejos, en los cuales se debe hacer referencia en el atributo *type* del elemento *message*, sin embargo en este trabajo no se tomaron en cuenta este tipo de elementos para mantener un estándar y

soportar los datos que la mayoría de los SABD contemplan y que son los tipos de datos definidos en el estándar SQL.

Tipos de puertos

Un tipo de puerto es un conjunto de operaciones abstractas y de mensajes abstractos que recibe un nombre, en la figura B.3 se presenta la parte del documento WSDL que corresponde a este elemento.

Ejemplo de Uso del Elemento portType

```
<wsdl:portType name="ServicioMySQL_port">
        <wsdl:operation name="Operacion1">
            <wsdl:input name="Operacion1Input" message="ServicioMySQL:Operacion1Input"/>
            <wsdl:output name="Operacion1Output" message="ServicioMySQL:Operacion1Output"/>
        </wsdl:operation>
</wsdl:portType>
```

Figura B.3 Elemento portType.

El atributo *name* de *portType* proporciona un nombre único entre todos los tipos de puertos definidos en el documento WSDL adjunto, en este caso *ServicioMySQL_port*.

Este elemento es una parte importante del documento WSDL ya que en el se definen las operaciones que el Servicio Web ofrece así como la relación con los elementos *message* que se crearon anteriormente, la relación entre el elemento *operacion* y el elemento *message* determina si son parámetros de entrada de la operación y por otro lado, como será el resultado arrojado por la ejecución de cada operación.

WSDL tiene cuatro primitivas de transmisión con las cuales las operaciones pueden estar ligadas según su funcionalidad:

- **Unidireccional:** El punto final recibe un mensaje.
- **Petición-respuesta:** El punto final recibe un mensaje y envía otro mensaje correlacionado.
- **Solicitud-respuesta:** El punto final envía un mensaje y recibe otro mensaje relacionado.
- **Notificación:** El punto final envía un mensaje.

WSDL hace referencia a estas primitivas como *operations*. Aunque las operaciones de petición-respuesta y solicitud-respuesta se pueden modelar de forma abstracta utilizando dos mensajes unidireccionales, resulta útil modelarlas como tipos de operaciones primitivas debido a que:

- Son muy comunes.
- La secuencia se puede ser correlacionar sin necesidad de introducir más información de flujo compleja.

- Algunos puntos finales sólo pueden recibir mensajes si son el resultado de una operación de petición-respuesta sincrónica.
- Un flujo simple puede derivarse algorítmicamente desde estas primitivas cuando se desee en la definición del flujo.

Cada tipo de operación se presenta en las secciones siguientes, sin embargo, cabe mencionar, que en este trabajo se eligió las operaciones de petición-respuesta y/o solicitud-respuesta debido a que debe informarse sobre los resultados arrojados por la ejecución de cada operación en cada Servicio Web, para saber si la ejecución fue exitosa u ocurrió algún tipo de error.

Operación unidireccional

La gramática para operaciones unidireccionales se muestra en la figura B.4.

Operación Unidireccional

```
<wsdl:definitions...> <wsdl:portType...> *
      <wsdl:operation name="nmtoken">
        <wsdl:input name="nmtoken"? message="qname"/>
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

Figura B.4 Operación Unidireccional.

El elemento *input* especifica el formato de mensaje abstracto para la operación unidireccional.

Operación de petición-respuesta

La gramática para operaciones de petición-respuesta se muestra en la figura B.5.

Operación Petición-Respuesta

```
<wsdl:definitions...>
   <wsdl:portType...> *
      <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
        <wsdl:input name="nmtoken"? message="qname"/>
        <wsdl:output name="nmtoken"? message="qname"/>
        <wsdl:fault name="nmtoken" message="qname"/>*
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

Figura B.5 Operación Petición-Respuesta.

Los elementos *input* y *output* especifican el formato de mensaje abstracto para la petición y la respuesta, respectivamente. Los elementos *fault* opcionales especifican el formato de mensaje abstracto para cualquier mensaje de error que pueda producirse como resultado de la operación (además de los específicos del protocolo).

Observe que la operación de petición-respuesta es abstracta; es necesario consultar un enlace concreto para determinar cómo se envían los mensajes dentro de una comunicación única (por ejemplo una petición/respuesta HTTP) o dos comunicaciones independientes (por ejemplo dos peticiones HTTP).

Operación de solicitud-respuesta

La gramática para operaciones de solicitud-respuesta se muestra en la figura B.6.

Operación Solicitud-Respuesta

```
<wsdl:definitions...>
   <wsdl:portType...> *
      <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
        <wsdl:output name="nmtoken"? message="qname"/>
        <wsdl:input name="nmtoken"? message="qname"/>
        <wsdl:fault name="nmtoken" message="qname"/>*
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

Figura B.6 Operación Solicitud-Respuesta.

Los elementos *input* y *output* especifican el formato de mensaje abstracto para la solicitud y la respuesta requeridas, respectivamente. Los elementos *fault* opcionales especifican el formato de mensaje abstracto para cualquier mensaje de error que pueda producirse como resultado de la operación (además de los específicos del protocolo).

Observe que la operación de petición-respuesta es abstracta; es necesario consultar un enlace concreto para determinar cómo se envían los mensajes dentro de una comunicación única (por ejemplo una petición/respuesta HTTP) o dos comunicaciones independientes (por ejemplo dos peticiones HTTP).

Operación de notificación

La gramática para operaciones de notificación se muestra en la figura B.7.

Operación Notificación

```
<wsdl:definitions...>
   <wsdl:portType...> *
      <wsdl:operation name="nmtoken">
         <wsdl:output name="nmtoken"? message="qname"/>
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

Figura B.7 Operación de Notificación.

El elemento *output* especifica el formato de mensaje abstracto para la operación de notificación.

Orden de los parámetros de una operación

Las operaciones no especifican si deben o no utilizarse con enlaces de tipo RPC. Sin embargo, cuando se utiliza una operación con un enlace RPC, conviene ser capaz de capturar la firma de la función de RPC original. Por esta razón, las operaciones de petición-respuesta o solicitud-respuesta pueden especificar una lista de nombres de parámetros mediante el atributo *parameterOrder* (de tipo *nmtokens*). El valor del atributo es una lista de nombres de las distintas partes del mensaje separados por un solo espacio. Las partes designadas deben seguir las reglas que se especifican a continuación:

- El orden en que se asignan nombres a las partes refleja el orden de los parámetros en la firma de RPC.
- La parte del valor RETURN no se presenta en la lista.
- Si un nombre de pieza aparece tanto en el mensaje entrante como en el saliente, se trata de un parámetro IN/OUT.
- Si un nombre de pieza aparece sólo en el mensaje entrante, se trata de un parámetro IN.
- Si un nombre de pieza aparece sólo en el mensaje saliente, se trata de un parámetro OUT.

Observe que esta información sirve de "sugerencia" y puede obviarse sin problema si no se está interesado en el tema de las firmas de RPC. Además, no se requiere estar presente, incluso si la operación se va a utilizar con un enlace de tipo RPC.

Por otro lado algunas implementaciones para consumir Servicios Web necesitan que exista este elemento para poder construir los mensajes SOAP enviados al servidor.

Enlaces

Los enlaces (binding) determinan el formato de mensaje y los detalles de protocolo de las operaciones y mensajes definidos por un *portType* determinado. Puede haber un número indeterminado de enlaces para un *portType* concreto. La gramática para enlaces se muestra en la figura B.8.

Elemento Enlace

```
<wsdl:binding name="ServicioMySQL_binding" type="ServicioMySQL:ServicioMySQL_port">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="Operacion1">
        <soap:operation soapAction="http://luis_movil:1010/WSDataBase/ServicioMySQL?Operacion1"/>
        <wsdl:input>
            <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SW4BD:ServicioMySQL"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:SW4BD:ServicioMySQL"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

Figura B.8 Ejemplo de Elemento Enlace.

El atributo *name* proporciona un nombre único entre todos los enlaces definidos en el documento WSDL adjunto, en este caso *ServicioMySQL_binding*.

Los enlaces hacen referencia al *portType* que enlazan utilizando el atributo *type*, en este caso *ServicioMySQL:ServicioMySQL_port*.

Las operaciones de enlace y los elementos de entrada, salida y error se correlacionan con los elementos *portType* correspondientes utilizando el atributo *name* de cada elemento y se comportan exactamente igual que en *portType*.

Los enlaces deben especificar exactamente un protocolo.

Los enlaces no pueden especificar información de dirección.

Puertos

Un puerto define un punto final de red individual especificando una dirección única para un enlace, su definición se muestra en la figura B.9.

Elemento Puerto

```
<wsdl:service name="ServicioMySQL_service">
    <wsdl:port name="ServicioMySQL_Port" binding="ServicioMySQL:ServicioMySQL_binding">
        <soap:address location="http://luis_movil:1010/WSDataBase/ServicioMySQL"/>
    </wsdl:port>
</wsdl:service>
```

Figura B.9 Ejemplo de Elemento Port.

El atributo *name* proporciona un nombre único entre todos los puertos definidos en el documento WSDL adjunto.

El atributo *binding* (de tipo QName) hace referencia al enlace que utiliza las reglas de vinculación definidas por WSDL.

Los elementos de extensibilidad del enlace, como *address* suelen especificar la información de dirección para el puerto.

Un puerto no puede especificar más de una dirección.

Un puerto no puede especificar ningún tipo de información de enlace distinta a la información de dirección.

Servicios

Un servicio agrupa un conjunto de puertos relacionados, en la figura B.10 se muestra un elemento Service:

Elemento Servicios

```
<wsdl:service name="ServicioMySQL_service">
    <wsdl:port name="ServicioMySQL_Port" binding="ServicioMySQL:ServicioMySQL_binding">
        <soap:address location="http://luis_movil:1010/WSDataBase/ServicioMySQL"/>
    </wsdl:port>
</wsdl:service>
```

Figura B.10 Ejemplo de Elemento Service.

El atributo *name* proporciona un nombre único entre todos los servicios definidos en el documento WSDL adjunto.

Los puertos de un servicio tienen las siguientes relaciones:

- Ningún puerto se comunica con los otros (por ejemplo, la salida de un puerto no es la entrada de otro).
- Si un servicio tiene varios puertos del mismo tipo, pero emplean enlaces o direcciones distintas, los puertos serán alternativas. Cada puerto presenta un comportamiento semánticamente equivalente (dentro de las limitaciones de transporte y formato de mensaje impuestas por cada enlace). Esto permite a un cliente de un documento WSDL elegir determinados puertos con los que comunicarse basándose en ciertos criterios (protocolo, distancia, etc.).
- Podemos determinar los tipos de puertos de un servicio mediante el examen de los mismos. Esto permite a un consumidor de un documento WSDL determinar si desea comunicarse con un servicio en particular, basándose en si admite o no distintos tipos de puertos. Esto resulta útil si existe alguna relación implícita entre las operaciones de los tipos de puertos y si todo el conjunto de tipos de puertos debe estar presente para cumplir una tarea determinada.

Como se mencionó al inicio de este anexo, existen otros elementos que pueden ser descritos dentro de un documento WSDL utilizando las extensiones y definiéndolos en base a esquemas de XML para un propósito específico, sin embargo este trabajo no contempla este tipo de elementos.

Anexo C
Scripts para crear Bases de Datos Ejemplo

En este anexo se presentan los Scripts que se utilizaron para la creación de la base de datos en cada SABD elegido, estos Script son los utilizados en el capítulo 6 para poder llevar a cabo las pruebas.

MySQL

Este Script corresponde al SABD MySQL el cual no soporta la creación de procedimientos almacenados ni funciones, además que la creación de vistas apenas es soportada en la versión 5.0 del SABD, sin embargo a pesar de que existe un controlador JDBC tipo 4 para este SABD, existe un problema de funcionamiento en él por lo que no es utilizado y no se crearon vistas.

El script es el siguiente:

Script SQL para MySQL

```
grant select, insert, update, delete
on tesis.*
to tesis@localhost identified by 'sw4bd';
drop database tesis;
CREATE DATABASE IF NOT EXISTS tesis;
use tesis;
DROP TABLE IF EXISTS Lis_x_mat;
DROP TABLE IF EXISTS FaltRet;
DROP TABLE IF EXISTS Clases;
DROP TABLE IF EXISTS alumnos;
DROP TABLE IF EXISTS maestros;
DROP TABLE IF EXISTS materias;
DROP TABLE IF EXISTS horarios;
DROP TABLE IF EXISTS salones;
create table alumnos (
      id_alumno int(5),
      nombre varchar(25),
      ape_pat varchar(25),
      ape_mat varchar(25),
      fec_nac date,
      fec_ing date,
      sexo char,
      CONSTRAINT PK_ALUMNO PRIMARY KEY (ID_ALUMNO)
);
create table maestros(
      id_maestro int(2),
      nombre varchar(25),
      ape_pat varchar(25),
      ape_mat varchar(25),
      grado char(1),
      fec_ing date,
```

```
      tipo char(1),
      CONSTRAINT PK_MAESTRO PRIMARY KEY (id_maestro)
);
create table materias(
      id_materia int(2),
      nombre varchar(40),
      CONSTRAINT PK_MATERIA PRIMARY KEY (id_materia)
);
create table horarios(
      id_horario char(5),
      descrip varchar(25),
      CONSTRAINT PK_HORARIO PRIMARY KEY (id_horario)
);
create table salones(
      id_salon char(5),
      descripcion varchar(25),
      CONSTRAINT PK_SALONES PRIMARY KEY (id_salon)
);
create table Clases(
      id_grupo int(3),
      id_materia int(2),
      id_maestro int(2),
      id_salon char(5),
      id_horario char(5),
      fec_ini date,
      dias char(5) not null,
      CONSTRAINT FK_CLAMAT foreign key (id_materia) REFERENCES MATERIAS,
      CONSTRAINT FK_CLASAL foreign key (id_salon) REFERENCES SALONES,
      CONSTRAINT FK_CLAMAE foreign key (id_maestro) REFERENCES MAESTROS,
      CONSTRAINT FK_CLAHOR foreign key (id_horario) REFERENCES HORARIOS,
      CONSTRAINT PK_CL PRIMARY KEY (ID_MATERIA, ID_MAESTRO, ID_SALON, ID_HORARIO)
);
create table Lis_x_mat(
      id_grupo int(3),
      id_alumno int(2),
      calif float(5,2),
      CONSTRAINT FK_LISMAT foreign key (id_grupo) REFERENCES Clases (id_grupo),
      CONSTRAINT FK_LISALU foreign key (id_alumno) REFERENCES ALUMNOS,
      CONSTRAINT PK_LM PRIMARY KEY (ID_GRUPO, ID_ALUMNO)
);
create table FaltRet(
      id_alumno int(5),
      id_grupo int(3),
      fecha date,
      tipo char(1),
      CONSTRAINT FK_FALALU foreign key (id_alumno) REFERENCES ALUMNOS,
      CONSTRAINT FK_FALMAT foreign key (id_grupo) REFERENCES CLASES,
      CONSTRAINT PK_FR PRIMARY KEY (ID_GRUPO, ID_ALUMNO)
);
insert into maestros values(1, 'Salvador', 'Aguilera', 'Sánchez', 'L',
str_to_date('01-01-1989','%d-%m-%Y'),'H');
insert into maestros values(2, 'José León', 'Cuevas', 'López', 'M',
str_to_date('01-02-1982','%d-%m-%Y'),'H');
insert into maestros values(3, 'Luis', 'Hernández', 'García', 'D',
str_to_date('01-03-1990','%d-%m-%Y'),'C');
insert into maestros values(4, 'Gonzalo', 'Muñiz', 'Rivera', 'L', str_to_date('01-
04-1978','%d-%m-%Y'),'H');
insert into maestros values(5, 'Patricia', 'Alvarado', 'Naranjo', 'M',
str_to_date('01-01-1992','%d-%m-%Y'),'H');
insert into maestros values(6, 'Antonia', 'Sandoval', 'Aguirre', 'D',
str_to_date('01-05-1995','%d-%m-%Y'),'C');
insert into maestros values(7, 'Francisco', 'Reyes', 'Cordero', 'L',
str_to_date('01-08-2000','%d-%m-%Y'),'H');
insert into maestros values(8, 'Karen', 'Peña', 'Rodríguez', 'M', str_to_date('01-
10-1989','%d-%m-%Y'),'H');
insert into maestros values(9, 'Cecilia', 'Mata', 'Morales', 'D', str_to_date('01-
12-1986','%d-%m-%Y'),'C');
insert into maestros values(10, 'Sandra', 'Gómez', 'Galícia', 'L',
str_to_date('01-01-1980','%d-%m-%Y'),'C');
commit;
insert into materias values (1, 'Programación Orientada a Objetos');
insert into materias values (2, 'Sistemas de Información');
insert into materias values (3, 'Introducción a la Computación');
insert into materias values (4, 'Diseño Lógico');
insert into materias values (5, 'Teoria de la Computación');
insert into materias values (6, 'Estructura de Datos');
insert into materias values (7, 'Arquitectura de Computadoras');
insert into materias values (8, 'Lenguajes de Programación');
insert into materias values (9, 'Base de Datos I');
```

```
insert into materias values (10, 'Programación de Sistemas I');
commit;
insert into salones values ('A_001','Edificio A segundo Piso');
insert into salones values ('A_002','Edificio A planta baja');
insert into salones values ('A_003','Edificio A primer piso');
insert into salones values ('A_004','Edificio A segundo Piso');
insert into salones values ('B_010','Edificio B');
insert into salones values ('B_011','Edificio B');
insert into salones values ('B_012','Edificio B');
insert into salones values ('B_001','Edificio B');
insert into salones values ('C_010','Edificio C planta baja');
insert into salones values ('C_001','Edificio C planta baja');
insert into salones values ('C_002','Laboratorio');
insert into salones values ('D_001','Laboratorio');
commit;
insert into horarios values ('M1_09','Matutino');
insert into horarios values ('M1_10','Matutino');
insert into horarios values ('M1_11','Matutino');
insert into horarios values ('M2_09','Matutino');
insert into horarios values ('M2_10','Matutino');
insert into horarios values ('M2_11','Matutino');
insert into horarios values ('T1_12','Vespertino');
insert into horarios values ('T1_13','Vespertino');
insert into horarios values ('T1_14','Vespertino');
insert into horarios values ('T2_14','Vespertino');
insert into horarios values ('T1_15','Vespertino');
insert into horarios values ('T1_16','Vespertino');
insert into horarios values ('T2_16','Nocturno');
insert into horarios values ('T1_17','Vespertino');
insert into horarios values ('T1_18','Vespertino');
insert into horarios values ('N1_19','Nocturno');
insert into horarios values ('N1_20','Nocturno');
commit;
insert into alumnos values (1, 'Franciso', 'López', 'Tovar', str_to_date('01-01-
1980','%d-%m-%Y'),str_to_date('01-01-2002','%d-%m-%Y'), 'M');
insert into alumnos values (2, 'Beatriz', 'González', 'Tello', str_to_date('22-02-
1982','%d-%m-%Y'),str_to_date('01-08-2000','%d-%m-%Y'), 'F');
insert into alumnos values (3, 'Pamela', 'Hernández', 'Lucio', str_to_date('14-11-
1983','%d-%m-%Y'),str_to_date('01-01-1998','%d-%m-%Y'), 'F');
insert into alumnos values (4, 'Gabriela', 'Esparza', 'Del Prado',
str_to_date('10-01-1984','%d-%m-%Y'),str_to_date('01-08-2002','%d-%m-%Y'), 'F');
insert into alumnos values (5, 'Ivett', 'Márquez', 'García', str_to_date('30-01-
1985','%d-%m-%Y'),str_to_date('01-01-2001','%d-%m-%Y'), 'F');
insert into alumnos values (6, 'Verónica', 'Correa', 'Álvarez', str_to_date('21-
05-1983','%d-%m-%Y'),str_to_date('01-08-2002','%d-%m-%Y'), 'F');
insert into alumnos values (7, 'Karla', 'Medina', 'Aguilar', str_to_date('09-07-
1984','%d-%m-%Y'),str_to_date('01-01-2003','%d-%m-%Y'), 'F');
insert into alumnos values (8, 'Jorge', 'Govea', 'Infante', str_to_date('03-06-
1982','%d-%m-%Y'),str_to_date('01-08-1998','%d-%m-%Y'), 'M');
insert into alumnos values (9, 'Mario', 'Villanueva', 'Rosado', str_to_date('14-
02-1981','%d-%m-%Y'),str_to_date('01-01-1999','%d-%m-%Y'), 'M');
insert into alumnos values (10, 'Jose Enrique', 'Gaviño', 'Castro',
str_to_date('01-04-1980','%d-%m-%Y'),str_to_date('01-08-1999','%d-%m-%Y'), 'M');
insert into alumnos values (11, 'Carlos', 'Barrera', 'Andrade', str_to_date('22-
03-1982','%d-%m-%Y'),str_to_date('01-01-2001','%d-%m-%Y'), 'M');
insert into alumnos values (12, 'Diana', 'Espinoza', 'Peña', str_to_date('18-08-
1983','%d-%m-%Y'),str_to_date('01-08-2001','%d-%m-%Y'), 'F');
insert into alumnos values (13, 'Arturo', 'Escalante', 'Lomelí', str_to_date('16-
10-1985','%d-%m-%Y'),str_to_date('01-01-2000','%d-%m-%Y'), 'M');
insert into alumnos values (14, 'Eircka', 'Cabrera', 'Landini', str_to_date('24-
12-1982','%d-%m-%Y'),str_to_date('01-08-2002','%d-%m-%Y'), 'F');
insert into alumnos values (15, 'Adrián', 'Aguirre', 'Fonseca', str_to_date('01-
09-1982','%d-%m-%Y'),str_to_date('01-01-2000','%d-%m-%Y'), 'M');
commit;
```

PostgreSQL

Este Script corresponde al SABD PostgreSQL, en este script se pusieron algunos elementos como vistas y como funciones para probar la creación del cuerpo de las operaciones, el script es el siguiente:

Script SQL para PostgreSQL

```
drop database tesis;
drop user tesis;
create user tesis with password 'sw4bd';
create database tesis owner tesis;
\connect tesis
create table alumnos (
      id_alumno int,
      nombre varchar(25),
      ape_pat varchar(25),
      ape_mat varchar(25),
      fec_nac date,
      fec_ing date,
      sexo char,
      CONSTRAINT PK_ALUMNO PRIMARY KEY (ID_ALUMNO)
);
create table maestros(
      id_maestro int,
      nombre varchar(25),
      ape_pat varchar(25),
      ape_mat varchar(25),
      grado char(1),
      fec_ing date,
      tipo char(1),
      CONSTRAINT PK_MAESTRO PRIMARY KEY (id_maestro)
);
create table materias(
      id_materia int,
      nombre varchar(40),
      CONSTRAINT PK_MATERIA PRIMARY KEY (id_materia)
);
create table horarios(
      id_horario char(5),
      descrip varchar(25),
      CONSTRAINT PK_HORARIO PRIMARY KEY (id_horario)
);
create table salones(
      id_salon char(5),
      descripcion varchar(25),
      CONSTRAINT PK_SALONES PRIMARY KEY (id_salon)
);
create table Clases(
      id_grupo int unique,
      id_materia int,
      id_maestro int,
      id_salon char(5),
      id_horario char(5),
      fec_ini date,
      dias char(5) not null,
      CONSTRAINT FK_CLAMAT foreign key (id_materia) REFERENCES MATERIAS,
      CONSTRAINT FK_CLASAL foreign key (id_salon) REFERENCES SALONES,
      CONSTRAINT FK_CLAMAE foreign key (id_maestro) REFERENCES MAESTROS,
      CONSTRAINT FK_CLAHOR foreign key (id_horario) REFERENCES HORARIOS,
      CONSTRAINT PK_CL PRIMARY KEY (ID_MATERIA, ID_MAESTRO, ID_SALON, ID_HORARIO)
);
create table Lis_x_mat(
           id_grupo int,
      id_alumno int,
      calif numeric(5,2),
      CONSTRAINT FK_LISMAT foreign key (id_grupo) REFERENCES CLASES(id_grupo),
      CONSTRAINT FK_LISALU foreign key (id_alumno) REFERENCES ALUMNOS,
      CONSTRAINT PK_LM PRIMARY KEY (ID_GRUPO, ID_ALUMNO)
);
create table FaltRet(
      id_alumno int,
      id_grupo int,
      fecha date,
      tipo char(1),
      CONSTRAINT FK_FALALU foreign key (id_alumno) REFERENCES ALUMNOS,
      CONSTRAINT FK_FALMAT foreign key (id_grupo) REFERENCES CLASES (id_grupo),
      CONSTRAINT PK_FR PRIMARY KEY (ID_GRUPO, ID_ALUMNO)
);
insert into maestros values(1, 'Salvador', 'Aguilera', 'Sánchez', 'L',
to_date('01-01-1989','dd-MM-yyyy'),'H');
insert into maestros values(2, 'José León', 'Cuevas', 'López', 'M', to_date('01-
02-1982','dd-MM-yyyy'),'H')
```

```
insert into maestros values(3, 'Luis', 'Hernández', 'García', 'D', to_date('01-03-
1990','dd-MM-yyyy'),'C');
insert into maestros values(4, 'Gonzalo', 'Muñiz', 'Rivera', 'L', to_date('01-04-
1978','dd-MM-yyyy'),'H');
insert into maestros values(5, 'Patricia', 'Alvarado', 'Naranjo', 'M',
to_date('01-01-1992','dd-MM-yyyy'),'H');
insert into maestros values(6, 'Antonia', 'Sandoval', 'Aguirre', 'D', to_date('01-
05-1995','dd-MM-yyyy'),'C');
insert into maestros values(7, 'Francisco', 'Reyes', 'Cordero', 'L', to_date('01-
08-2000','dd-MM-yyyy'),'H');
insert into maestros values(8, 'Karen', 'Peña', 'Rodríguez', 'M', to_date('01-10-
1989','dd-MM-yyyy'),'H');
insert into maestros values(9, 'Cecilia', 'Mata', 'Morales', 'D', to_date('01-12-
1986','dd-MM-yyyy'),'C');
insert into maestros values(10, 'Sandra', 'Gómez', 'Galícia', 'L', to_date('01-01-
1980','dd-MM-yyyy'),'C');
commit;
insert into materias values (1, 'Programación Orientada a Objetos');
insert into materias values (2, 'Sistemas de Información');
insert into materias values (3, 'Introducción a la Computación');
insert into materias values (4, 'Diseño Lógico');
insert into materias values (5, 'Teoria de la Computación');
insert into materias values (6, 'Estructura de Datos');
insert into materias values (7, 'Arquitectura de Computadoras');
insert into materias values (8, 'Lenguajes de Programación');
insert into materias values (9, 'Base de Datos I');
insert into materias values (10, 'Programación de Sistemas I');
commit;
insert into salones values ('A_001','Edificio A segundo Piso');
insert into salones values ('A_002','Edificio A planta baja');
insert into salones values ('A_003','Edificio A primer piso');
insert into salones values ('A_004','Edificio A segundo Piso');
insert into salones values ('B_010','Edificio B');
insert into salones values ('B_011','Edificio B');
insert into salones values ('B_012','Edificio B');
insert into salones values ('B_001','Edificio B');
insert into salones values ('C_010','Edificio C planta baja');
insert into salones values ('C_001','Edificio C planta baja');
insert into salones values ('C_002','Laboratorio');
insert into salones values ('D_001','Laboratorio');
commit;
insert into horarios values ('M1_09','Matutino');
insert into horarios values ('M1_10','Matutino');
insert into horarios values ('M1_11','Matutino');
insert into horarios values ('M2_09','Matutino');
insert into horarios values ('M2_10','Matutino');
insert into horarios values ('M2_11','Matutino');
insert into horarios values ('T1_12','Vespertino');
insert into horarios values ('T1_13','Vespertino');
insert into horarios values ('T1_14','Vespertino');
insert into horarios values ('T2_14','Vespertino');
insert into horarios values ('T1_15','Vespertino');
insert into horarios values ('T1_16','Vespertino');
insert into horarios values ('T2_16','Nocturno');
insert into horarios values ('T1_17','Vespertino');
insert into horarios values ('T1_18','Vespertino');
insert into horarios values ('N1_19','Nocturno');
insert into horarios values ('N1_20','Nocturno');
commit;
insert into alumnos values (1, 'Franciso', 'López', 'Tovar', to_date('01-01-
1980','dd-MM-yyyy'),to_date('01-01-2002','dd-MM-yyyy'), 'M');
insert into alumnos values (2, 'Beatriz', 'González', 'Tello', to_date('22-02-
1982','dd-MM-yyyy'),to_date('01-08-2000','dd-MM-yyyy'), 'F');
insert into alumnos values (3, 'Pamela', 'Hernández', 'Lucio', to_date('14-11-
1983','dd-MM-yyyy'),to_date('01-01-1998','dd-MM-yyyy'), 'F');
insert into alumnos values (4, 'Gabriela', 'Esparza', 'Del Prado', to_date('10-01-
1984','dd-MM-yyyy'),to_date('01-08-2002','dd-MM-yyyy'), 'F');
insert into alumnos values (5, 'Ivett', 'Márquez', 'García', to_date('30-01-
1985','dd-MM-yyyy'),to_date('01-01-2001','dd-MM-yyyy'), 'F');
insert into alumnos values (6, 'Verónica', 'Correa', 'Álvarez', to_date('21-05-
1983','dd-MM-yyyy'),to_date('01-08-2002','dd-MM-yyyy'), 'F');
insert into alumnos values (7, 'Karla', 'Medina', 'Aguilar', to_date('09-07-
1984','dd-MM-yyyy'),to_date('01-01-2003','dd-MM-yyyy'), 'F');
insert into alumnos values (8, 'Jorge', 'Govea', 'Infante', to_date('03-06-
1982','dd-MM-yyyy'),to_date('01-08-1998','dd-MM-yyyy'), 'M');
insert into alumnos values (9, 'Mario', 'Villanueva', 'Rosado', to_date('14-02-
1981','dd-MM-yyyy'),to_date('01-01-1999','dd-MM-yyyy'), 'M');
insert into alumnos values (10, 'Jose Enrique', 'Gaviño', 'Castro', to_date('01-
04-1980','dd-MM-yyyy'),to_date('01-08-1999','dd-MM-yyyy'), 'M');
```

```
insert into alumnos values (11, 'Carlos', 'Barrera', 'Andrade', to_date('22-03-
1982','dd-MM-yyyy'),to_date('01-01-2001','dd-MM-yyyy'), 'M');
insert into alumnos values (12, 'Diana', 'Espinoza', 'Peña', to_date('18-08-
1983','dd-MM-yyyy'),to_date('01-08-2001','dd-MM-yyyy'), 'F');
insert into alumnos values (13, 'Arturo', 'Escalante', 'Lomelí', to_date('16-10-
1985','dd-MM-yyyy'),to_date('01-01-2000','dd-MM-yyyy'), 'M');
insert into alumnos values (14, 'Eircka', 'Cabrera', 'Landini', to_date('24-12-
1982','dd-MM-yyyy'),to_date('01-08-2002','dd-MM-yyyy'), 'F');
insert into alumnos values (15, 'Adrián', 'Aguirre', 'Fonseca', to_date('01-09-
1982','dd-MM-yyyy'),to_date('01-01-2000','dd-MM-yyyy'), 'M');
commit;
---- Vistas
DROP VIEW listxgrupo;
DROP VIEW matexalumn;
DROP VIEW retfalxalu;
DROP VIEW matexmaestro;
DROP VIEW retfalxmateria;
CREATE VIEW listxgrupo
AS
   SELECT cl.id_grupo as id_grupo, cl.id_materia AS idmat, mat.nombre AS nommat,
          al.id_alumno AS idalu, al.nombre AS nomalu, al.ape_pat as aluapp,
al.ape_mat as aluapm, cl.id_maestro idpro,
          prf.nombre AS nompro
     FROM lis_x_mat lm, alumnos al, materias mat, clases cl, maestros prf
    WHERE lm.id_grupo = cl.id_grupo
      AND cl.id_materia = mat.id_materia
      AND cl.id_maestro = prf.id_maestro
      AND lm.id_alumno = al.id_alumno
 ORDER BY idmat, idalu;
CREATE VIEW matexalumn
AS
   SELECT al.id_alumno AS idalu, al.nombre AS nomalu, al.ape_pat as aluapp,
al.ape_mat as aluapm, mat.id_materia AS idmat, mat.nombre AS nommat
     FROM lis_x_mat lm, alumnos al, materias mat, clases cl
    WHERE lm.id_grupo = cl.id_grupo
      AND lm.id_alumno = al.id_alumno
      AND cl.id_materia = mat.id_materia
 ORDER BY idalu, idmat;
CREATE VIEW retfalxalu
AS
   SELECT res.alu AS idalu, al.nombre AS nomalu, al.ape_pat as aluapp, al.ape_mat
as aluapm, mat.id_materia AS idmat, mat.nombre AS nommat,
          res.numret AS numret, res.numfal AS numfal
     FROM (SELECT   fr.id_grupo, fr.id_alumno AS alu,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                           END) AS numfal,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                           END) AS numret
               FROM faltret fr
           GROUP BY fr.id_alumno, fr.id_grupo) res,
          alumnos al,
          materias mat,
          clases cl
    WHERE cl.id_materia = mat.id_materia
      AND res.alu = al.id_alumno
      AND res.id_grupo = cl.id_grupo
 ORDER BY idalu, idmat;
CREATE VIEW matexmaestro
AS
   SELECT prf.id_maestro AS idprf, prf.nombre AS nomprf, prf.ape_pat AS appprf,
prf.ape_mat AS apmprf,
          mat.id_materia AS idmat, mat.nombre AS nommat
     FROM maestros prf, materias mat, clases cl
    WHERE prf.id_maestro = cl.id_maestro AND cl.id_materia = mat.id_materia
 ORDER BY idprf, idmat;
CREATE VIEW retfalxmateria
AS
   SELECT cl.id_grupo AS idgrp, mat.id_materia AS idmat, mat.nombre as nommat,
          res.numret AS numret, res.numfal AS numfal
     FROM (SELECT   fr.id_grupo,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
```

```
                              ELSE 0
                          END) AS numfal,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                          END) AS numret
               FROM faltret fr
           GROUP BY fr.id_grupo) res,
          materias mat,
          clases cl
    WHERE cl.id_materia = mat.id_materia
      AND res.id_grupo = cl.id_grupo
 ORDER BY idmat;
-------- Procedimientos
drop function mov_alumnos(int, varchar(25), varchar(25), varchar(25), date, date,
char, int);
drop function mov_maestros(int, varchar(25), varchar(25), varchar(25), char, date,
char, int);
drop function mov_materias(int, varchar(25), int);
drop function mov_salones(char(5), varchar(25), int);
drop function mov_horarios(char(5), varchar(25), int);

create function mov_alumnos(int, varchar(25), varchar(25), varchar(25), date,
date, char, int) returns int as '
declare
   vid_al alias for $1;
   vnom alias for $2;
   vapp alias for $3;
   vapm alias for $4;
   vfen alias for $5;
   vfei alias for $6;
   vsex alias for $7;
   vtip alias for $8;
   vret int;
   vult int;
begin
   vret:=0;
   if vtip = 0 then
      select max(id_alumno)+1 into vult from alumnos;
      insert into alumnos values (vult, vnom, vapp, vapm, vfen, vfei, vsex);
      select max(id_alumno) into vret from alumnos;
      if not found or vult>vret then
         vret:=-1;
      else
         vret:=vult;
      end if;
   end if;
   if vtip = 1 then
      select id_alumno into vret from alumnos where id_alumno = vid_al;
      if not found then
         vret:=-1;
      else
         update alumnos set nombre = vnom, ape_pat = vapp, ape_mat= vapm, fec_nac
= vfen, fec_ing = vfei, sexo = vsex where id_alumno = vid_al;
         vret:=0;
      end if;
   end if;
   if vtip = 2 then
      select id_alumno into vret from alumnos where id_alumno = vid_al;
      if not found then
         vret:=-1;
      else
         delete from alumnos where id_alumno = vid_al;
         vret :=0;
      end if;
   end if;
   return(vret);
end;
' language 'plpgsql';
create function mov_maestros(int, varchar(25), varchar(25), varchar(25), char,
date, char, int) returns int as '
declare
   vid_ma alias for $1;
   vnom alias for $2;
   vapp alias for $3;
   vapm alias for $4;
   vgrm alias for $5;
   vfei alias for $6;
```

```
    vtma alias for $7;
    vtip alias for $8;
    vret int;
    vult int;
begin
    vret:=0;
    if vtip = 0 then
       select max(id_maestro)+1 into vult from maestros;
       insert into maestros values (vult, vnom, vapp, vapm, vgrm, vfei, vtma);
       select max(id_maestro) into vret from maestros;
       if not found or vult>vret then
          vret:=-1;
       else
          vret:=vult;
       end if;
    end if;
    if vtip = 1 then
       select id_maestro into vret from maestros where id_maestro = vid_ma;
       if not found then
          vret:=-1;
       else
          update maestros set nombre = vnom, ape_pat = vapp, ape_mat= vapm, grado =
vgrm, fec_ing = vfen, tipo = vtma where id_maestro = vid_ma;
          vret:=0;
       end if;
    end if;
    if vtip = 2 then
       select id_maestro into vret from maestros where id_maestro = vid_ma;
       if not found then
          vret:=-1;
       else
          delete from maestros where id_maestro = vid_ma;
          vret :=0;
       end if;
    end if;
    return(vret);

end;
' language 'plpgsql';
create function mov_materias(int, varchar(40), int) returns int as '
declare
    vid_ma alias for $1;
    vnom alias for $2;
    vtip alias for $3;
    vret int;
    vult int;
begin
    vret:=0;
    if vtip = 0 then
       select max(id_materia)+1 into vult from materias;
       insert into materias values (vult, vnom);
       select max(id_materia) into vret from materias;
       if not found or vult>vret then
          vret:=-1;
       else
          vret:=vult;
       end if;
    end if;
    if vtip = 1 then
       select id_materia into vret from materias where id_materia = vid_ma;
       if not found then
          vret:=-1;
       else
          update materias set nombre = vnom where id_materia = vid_ma;
          vret:=0;
       end if;
    end if;
    if vtip = 2 then
       select id_materia into vret from materias where id_materia = vid_ma;
       if not found then
          vret:=-1;
       else
          delete from materias where id_materia = vid_ma;
          vret :=0;
       end if;
    end if;
    return(vret);

end;
```

```
' language 'plpgsql';
create function mov_salones(char(5), varchar(25), int) returns int as '
declare
   vid_sa alias for $1;
   vnom alias for $2;
   vtip alias for $3;
   vret int;
   vult int;
   vval char(5);
begin
   vret:=0;
   if vtip = 0 then
      insert into salones values (vid_sa, vnom);
      select id_salon into vval from salones where id_salon = vid_sa;
      if not found then
         vret:=-1;
      else
         vret:=0;
      end if;
   end if;
   if vtip = 1 then
      select id_salon into vval from salones where id_salon = vid_sa;
      if not found then
         vret:=-1;
      else
         update salones set descripcion = vnom where id_salon = vid_sa;
         vret:=0;
      end if;
   end if;
   if vtip = 2 then
      select id_salon into vval from salones where id_salon = vid_sa;
      if not found then
         vret:=-1;
      else
         delete from salones where id_salon = vid_sa;
         vret :=0;
      end if;
   end if;
   return(vret);

end;
' language 'plpgsql';
create function mov_horarios(char(5), varchar(25), int) returns int as '
declare
   vid_hr alias for $1;
   vnom alias for $2;
   vtip alias for $3;
   vret int;
   vult int;
   vval char(5);
begin
   vret:=0;
   if vtip = 0 then
      insert into horarios values (vid_hr, vnom);
      select id_horario into vval from horarios where id_horario = vid_hr;
      if not found then
         vret:=-1;
      else
         vret:=0;
      end if;
   end if;
   if vtip = 1 then
      select id_horario into vval from horarios where id_horario = vid_hr;
      if not found then
         vret:=-1;
      else
         update horarios set descrip = vnom where id_horario = vid_hr;
         vret:=0;
      end if;
   end if;
   if vtip = 2 then
      select id_horario into vval from horarios where id_horario = vid_hr;
      if not found then
         vret:=-1;
      else
         delete from horarios where id_horario = vid_hr;
         vret :=0;
      end if;
   end if;
```

```
   return(vret);
end;
' language 'plpgsql';
```

Oracle

Este Script corresponde al SABD Oracle y fue elegido porque es el que presenta características suficientes para poder probar las funciones que ofrece esta aplicación:

Script SQL para Oracle

```
GRANT CONNECT,RESOURCE,UNLIMITED TABLESPACE TO TESIS IDENTIFIED BY sw4bd;
ALTER USER TESIS DEFAULT TABLESPACE USERS;
ALTER USER TESIS TEMPORARY TABLESPACE TEMP;
CONNECT TESIS/sw4bd;
DROP TABLE Lis_x_mat;
DROP TABLE FaltRet;
DROP TABLE Clases;
DROP TABLE alumnos;
DROP TABLE maestros;
DROP TABLE materias;
DROP TABLE horarios;
DROP TABLE salones;
create table alumnos (
      id_alumno number(5) CONSTRAINT PK_ALUMNO PRIMARY KEY,
      nombre varchar2(25),
      ape_pat varchar2(25),
      ape_mat varchar2(25),
      fec_nac date,
      fec_ing date,
      sexo char CONSTRAINT ALU_SEX check (sexo in ('M','F'))
);
create table maestros(
      id_maestro number(2) CONSTRAINT PK_MAESTRO PRIMARY KEY,
      nombre varchar2(25),
      ape_pat varchar2(25),
      ape_mat varchar2(25),
      grado char(1) CONSTRAINT CMAE_GRA check (grado in ('L','M','D')),
      fec_ing date,
      tipo char(1) CONSTRAINT CMAE_TIP check (tipo in ('H','C'))
);
create table materias(
      id_materia number(2) CONSTRAINT PK_MATERIA PRIMARY KEY,
      nombre varchar2(40)
);
create table horarios(
      id_horario char(5) CONSTRAINT PK_HORARIO PRIMARY KEY,
      descrip varchar2(25)
);
create table salones(
      id_salon char(5) CONSTRAINT PK_SALONES PRIMARY KEY,
      descripcion varchar2(25)
);
create table Clases(
      id_grupo number(3) unique,
      id_materia number(2) CONSTRAINT FK_CLAMAT REFERENCES MATERIAS,
      id_maestro number(2) CONSTRAINT FK_CLAMAE REFERENCES MAESTROS,
      id_salon char(5) CONSTRAINT FK_CLASAL REFERENCES SALONES,
      id_horario char(5) CONSTRAINT FK_CLAHOR REFERENCES HORARIOS,
      fec_ini date,
      dias char(5) not null
```

```
);
create table Lis_x_mat(
      id_grupo number(3) CONSTRAINT FK_LISMAT REFERENCES CLASES
(id_grupo),
      id_alumno number(2) CONSTRAINT FK_LISALU REFERENCES ALUMNOS,
      calif number(5,2)
);
create table FaltRet(
      id_alumno number(5) CONSTRAINT FK_FALALU REFERENCES ALUMNOS,
      id_grupo number(3) CONSTRAINT FK_FALMAT REFERENCES CLASES
(id_grupo),
      fecha date,
      tipo char(1) CONSTRAINT FR_TIP check (tipo in ('F','R'))
);
ALTER TABLE LIS_X_MAT ADD (CONSTRAINT PK_LM PRIMARY KEY (ID_GRUPO,
ID_ALUMNO));
ALTER TABLE FALTRET ADD (CONSTRAINT PK_FR PRIMARY KEY (ID_GRUPO,
ID_ALUMNO));
ALTER TABLE CLASES ADD (CONSTRAINT PK_CL PRIMARY KEY (ID_MATERIA,
ID_MAESTRO, ID_SALON, ID_HORARIO));
insert into maestros values(1, 'Salvador', 'Aguilera', 'Sánchez', 'L',
to_date('01-01-1989','dd-MM-yyyy'),'H');
insert into maestros values(2, 'José León', 'Cuevas', 'López', 'M',
to_date('01-02-1982','dd-MM-yyyy'),'H');
insert into maestros values(3, 'Luis', 'Hernández', 'García', 'D',
to_date('01-03-1990','dd-MM-yyyy'),'C');
insert into maestros values(4, 'Gonzalo', 'Muñiz', 'Rivera', 'L',
to_date('01-04-1978','dd-MM-yyyy'),'H');
insert into maestros values(5, 'Patricia', 'Alvarado', 'Naranjo', 'M',
to_date('01-01-1992','dd-MM-yyyy'),'H');
insert into maestros values(6, 'Antonia', 'Sandoval', 'Aguirre', 'D',
to_date('01-05-1995','dd-MM-yyyy'),'C');
insert into maestros values(7, 'Francisco', 'Reyes', 'Cordero', 'L',
to_date('01-08-2000','dd-MM-yyyy'),'H');
insert into maestros values(8, 'Karen', 'Peña', 'Rodríguez', 'M',
to_date('01-10-1989','dd-MM-yyyy'),'H');
insert into maestros values(9, 'Cecilia', 'Mata', 'Morales', 'D',
to_date('01-12-1986','dd-MM-yyyy'),'C');
insert into maestros values(10, 'Sandra', 'Gómez', 'Galícia', 'L',
to_date('01-01-1980','dd-MM-yyyy'),'C');
commit;
insert into materias values (1, 'Programación Orientada a Objetos');
insert into materias values (2, 'Sistemas de Información');
insert into materias values (3, 'Introducción a la Computación');
insert into materias values (4, 'Diseño Lógico');
insert into materias values (5, 'Teoria de la Computación');
insert into materias values (6, 'Estructura de Datos');
insert into materias values (7, 'Arquitectura de Computadoras');
insert into materias values (8, 'Lenguajes de Programación');
insert into materias values (9, 'Base de Datos I');
insert into materias values (10, 'Programación de Sistemas I');
commit;
insert into salones values ('A_001','Edificio A segundo Piso');
insert into salones values ('A_002','Edificio A planta baja');
insert into salones values ('A_003','Edificio A primer piso');
insert into salones values ('A_004','Edificio A segundo Piso');
insert into salones values ('B_010','Edificio B');
insert into salones values ('B_011','Edificio B');
insert into salones values ('B_012','Edificio B');
insert into salones values ('B_001','Edificio B');
insert into salones values ('C_010','Edificio C planta baja');
insert into salones values ('C_001','Edificio C planta baja');
insert into salones values ('C_002','Laboratorio');
insert into salones values ('D_001','Laboratorio');
commit;
insert into horarios values ('M1_09','Matutino');
insert into horarios values ('M1_10','Matutino');
insert into horarios values ('M1_11','Matutino');
insert into horarios values ('M2_09','Matutino');
insert into horarios values ('M2_10','Matutino');
```

```
insert into horarios values ('M2_11','Matutino');
insert into horarios values ('T1_12','Vespertino');
insert into horarios values ('T1_13','Vespertino');
insert into horarios values ('T1_14','Vespertino');
insert into horarios values ('T2_14','Vespertino');
insert into horarios values ('T1_15','Vespertino');
insert into horarios values ('T1_16','Vespertino');
insert into horarios values ('T2_16','Nocturno');
insert into horarios values ('T1_17','Vespertino');
insert into horarios values ('T1_18','Vespertino');
insert into horarios values ('N1_19','Nocturno');
insert into horarios values ('N1_20','Nocturno');
commit;
insert into alumnos values (1, 'Franciso', 'López', 'Tovar', to_date('01-
01-1980','dd-MM-yyyy'),to_date('01-01-2002','dd-MM-yyyy'), 'M');
insert into alumnos values (2, 'Beatriz', 'González', 'Tello',
to_date('22-02-1982','dd-MM-yyyy'),to_date('01-08-2000','dd-MM-yyyy'),
'F');
insert into alumnos values (3, 'Pamela', 'Hernández', 'Lucio',
to_date('14-11-1983','dd-MM-yyyy'),to_date('01-01-1998','dd-MM-yyyy'),
'F');
insert into alumnos values (4, 'Gabriela', 'Esparza', 'Del Prado',
to_date('10-01-1984','dd-MM-yyyy'),to_date('01-08-2002','dd-MM-yyyy'),
'F');
insert into alumnos values (5, 'Ivett', 'Márquez', 'García', to_date('30-
01-1985','dd-MM-yyyy'),to_date('01-01-2001','dd-MM-yyyy'), 'F');
insert into alumnos values (6, 'Verónica', 'Correa', 'Álvarez',
to_date('21-05-1983','dd-MM-yyyy'),to_date('01-08-2002','dd-MM-yyyy'),
'F');
insert into alumnos values (7, 'Karla', 'Medina', 'Aguilar', to_date('09-
07-1984','dd-MM-yyyy'),to_date('01-01-2003','dd-MM-yyyy'), 'F');
insert into alumnos values (8, 'Jorge', 'Govea', 'Infante', to_date('03-
06-1982','dd-MM-yyyy'),to_date('01-08-1998','dd-MM-yyyy'), 'M');
insert into alumnos values (9, 'Mario', 'Villanueva', 'Rosado',
to_date('14-02-1981','dd-MM-yyyy'),to_date('01-01-1999','dd-MM-yyyy'),
'M');
insert into alumnos values (10, 'Jose Enrique', 'Gaviño', 'Castro',
to_date('01-04-1980','dd-MM-yyyy'),to_date('01-08-1999','dd-MM-yyyy'),
'M');
insert into alumnos values (11, 'Carlos', 'Barrera', 'Andrade',
to_date('22-03-1982','dd-MM-yyyy'),to_date('01-01-2001','dd-MM-yyyy'),
'M');
insert into alumnos values (12, 'Diana', 'Espinoza', 'Peña', to_date('18-
08-1983','dd-MM-yyyy'),to_date('01-08-2001','dd-MM-yyyy'), 'F');
insert into alumnos values (13, 'Arturo', 'Escalante', 'Lomelí',
to_date('16-10-1985','dd-MM-yyyy'),to_date('01-01-2000','dd-MM-yyyy'),
'M');
insert into alumnos values (14, 'Eircka', 'Cabrera', 'Landini',
to_date('24-12-1982','dd-MM-yyyy'),to_date('01-08-2002','dd-MM-yyyy'),
'F');
insert into alumnos values (15, 'Adrián', 'Aguirre', 'Fonseca',
to_date('01-09-1982','dd-MM-yyyy'),to_date('01-01-2000','dd-MM-yyyy'),
'M');
commit;
---- Vistas
DROP VIEW listxgrupo;
DROP VIEW matexalumn;
DROP VIEW retfalxalu;
DROP VIEW matexmaestro;
DROP VIEW retfalxmateria
CREATE VIEW listxgrupo
AS
   SELECT cl.id_grupo as id_grupo, cl.id_materia AS idmat, mat.nombre AS
nommat,
          al.id_alumno AS idalu, al.nombre AS nomalu, al.ape_pat as
aluapp, al.ape_mat as aluapm, cl.id_maestro idpro,
          prf.nombre AS nompro
     FROM lis_x_mat lm, alumnos al, materias mat, clases cl, maestros prf
    WHERE lm.id_grupo = cl.id_grupo
      AND cl.id_materia = mat.id_materia
```

```
      AND cl.id_maestro = prf.id_maestro
      AND lm.id_alumno = al.id_alumno
 ORDER BY idmat, idalu;
CREATE VIEW matexalumn
AS
   SELECT al.id_alumno AS idalu, al.nombre AS nomalu, al.ape_pat as
aluapp, al.ape_mat as aluapm, mat.id_materia AS idmat, mat.nombre AS
nommat
     FROM lis_x_mat lm, alumnos al, materias mat, clases cl
    WHERE lm.id_grupo = cl.id_grupo
      AND lm.id_alumno = al.id_alumno
      AND cl.id_materia = mat.id_materia
 ORDER BY idalu, idmat;
CREATE VIEW retfalxalu
AS
   SELECT res.alu AS idalu, al.nombre AS nomalu, al.ape_pat as aluapp,
al.ape_mat as aluapm, mat.id_materia idmat, mat.nombre as nommat,
          res.numret AS numret, res.numfal AS numfal
     FROM (SELECT   fr.id_grupo, fr.id_alumno alu,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                           END) numfal,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                           END) numret
               FROM faltret fr
           GROUP BY fr.id_alumno, fr.id_grupo) res,
          alumnos al,
          materias mat,
          clases cl
    WHERE cl.id_materia = mat.id_materia
      AND res.alu = al.id_alumno
      AND res.id_grupo = cl.id_grupo
 ORDER BY idalu, idmat;
CREATE VIEW matexmaestro
AS
   SELECT prf.id_maestro AS idprf, prf.nombre AS nomprf, prf.ape_pat AS
appprf, prf.ape_mat AS apmprf,
          mat.id_materia AS idmat, mat.nombre AS nommat
     FROM maestros prf, materias mat, clases cl
    WHERE prf.id_maestro = cl.id_maestro AND cl.id_materia =
mat.id_materia
 ORDER BY idprf, idmat;
CREATE VIEW retfalxmateria
AS
   SELECT cl.id_grupo idgrp, mat.id_materia idmat, mat.nombre as nommat,
          res.numret AS numret, res.numfal AS numfal
     FROM (SELECT   fr.id_grupo,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                           END) numfal,
                    COUNT (CASE
                              WHEN tipo = '1'
                                 THEN 1
                              ELSE 0
                           END) numret
               FROM faltret fr
           GROUP BY fr.id_grupo) res,
          materias mat,
          clases cl
    WHERE cl.id_materia = mat.id_materia
      AND res.id_grupo = cl.id_grupo
 ORDER BY idmat;
-------- Procedimientos
```

```
CREATE OR REPLACE procedure mov_alumnos(vid_al number, vnom varchar2,
vapp varchar2, vapm varchar2,
                                vfen date, vfei date, vsex char, vtip
number, vret out number) as
   vult int;
begin
   vret:=0;
   if vtip = 0 then
      select max(id_alumno)+1 into vult from alumnos;
      insert into alumnos values (vult, vnom, vapp, vapm, vfen, vfei,
vsex);
      select max(id_alumno) into vret from alumnos;
      if SQL%NOTFOUND or vult>vret then
         vret:=-1;
      else
         vret:=vult;
      end if;
   end if;
   if vtip = 1 then
      if not vid_al is null then
       update alumnos set nombre = vnom, ape_pat = vapp, ape_mat= vapm,
fec_nac = vfen, fec_ing = vfei, sexo = vsex where id_alumno = vid_al;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
      end if;
       else
            vret:=-1;
     end if;
   end if;
   if vtip = 2 then
      if not vid_al is null then
      delete from alumnos where id_alumno = vid_al;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret :=0;
      end if;
       else
          vret:=-1;
       end if;
   end if;
end;
CREATE OR REPLACE procedure mov_maestros(vid_ma number, vnom varchar2,
vapp varchar2, vapm varchar2,
                                vgrm char, vfei date, vtma char, vtip
number, vret out number) as
   vult int;
begin
   vret:=0;
   if vtip = 0 then
      select max(id_maestro)+1 into vult from maestros;
      insert into maestros values (vult, vnom, vapp, vapm, vgrm, vfei,
vtma);
      select max(id_maestro) into vret from maestros;
      if SQL%NOTFOUND or vult>vret then
         vret:=-1;
      else
         vret:=vult;
      end if;
   end if;
   if vtip = 1 then
      if not vid_ma is null then
       update maestros set nombre = vnom, ape_pat = vapp, ape_mat= vapm,
grado = vgrm, fec_ing = vfei, tipo = vtma where id_maestro = vid_ma;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
```

```
      end if;
      else
            vret:=-1;
     end if;
   end if;
   if vtip = 2 then
      if not vid_ma is null then
      delete from maestros where id_maestro = vid_ma;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret :=0;
      end if;
      else
         vret:=-1;
      end if;
   end if;
end;
CREATE OR REPLACE procedure mov_materias(vid_ma number, vnom varchar2,
vtip number, vret out number) as
   vult int;
begin
   vret:=0;
   if vtip = 0 then
      select max(id_materia)+1 into vult from materias;
      insert into materias values (vult, vnom);
      select max(id_materia) into vret from materias;
      if SQL%NOTFOUND or vult>vret then
         vret:=-1;
      else
         vret:=vult;
      end if;
   end if;
   if vtip = 1 then
      if not vid_ma is null then
       update materias set nombre = vnom where id_materia = vid_ma;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
      end if;
      else
            vret:=-1;
     end if;
   end if;
   if vtip = 2 then
      if not vid_ma is null then
      delete from materias where id_materia = vid_ma;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret :=0;
      end if;
      else
         vret:=-1;
      end if;
   end if;
end;
CREATE OR REPLACE procedure mov_salones(vid_sa char, vnom varchar2, vtip
number, vret out number) as
   vult char(5);
begin
   vret:=0;
   if vtip = 0 then
      insert into salones values (vid_sa, vnom);
      select id_salon into vult from salones where id_salon = vid_sa;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
```

```
      end if;
   end if;
   if vtip = 1 then
      if not vid_sa is null then
       update salones set descripcion = vnom where id_salon = vid_sa;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
      end if;
      else
           vret:=-1;
     end if;
   end if;
   if vtip = 2 then
      if not vid_sa is null then
      delete from salones where id_salon = vid_sa;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret :=0;
      end if;
      else
         vret:=-1;
      end if;
   end if;
end;
CREATE OR REPLACE procedure mov_horarios(vid_hr char, vnom varchar2, vtip
number, vret out number) as
   vult char(5);
begin
   vret:=0;
   if vtip = 0 then
      insert into horarios values (vid_hr, vnom);
      select id_horario into vult from horarios where id_horario=vid_hr;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
      end if;
   end if;
   if vtip = 1 then
      if not vid_hr is null then
       update horarios set descrip = vnom where id_horario = vid_hr;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret:=0;
      end if;
      else
           vret:=-1;
     end if;
   end if;
   if vtip = 2 then
      if not vid_hr is null then
      delete from horarios where id_horario = vid_hr;
      if SQL%NOTFOUND then
         vret:=-1;
      else
         vret :=0;
      end if;
      else
         vret:=-1;
      end if;
   end if;
end;
```

Anexo D
Manual

Instalación

Para poder llevar a cabo la instalación de esta aplicación, es necesario contar con los siguientes requerimientos:

- Contar con el JRE (Java Runtime Environment) en su versión 1.4 o superior.
- Cumplir con los requisitos de instalación del JRE.
- Se recomienda 15 MB de espacio libre en disco duro.

En caso de no contar con el JRE de Java, en el directorio JavaJRE se encuentran los archivos de instalación para la versión 1.4.

Una vez cumplido con estos requisitos de instalación, deberá ejecutar el archivo de instalación correspondiente a su plataforma (Linux, Windows, etc.), los archivos de instalación vienen en el CD-ROM que compaña a este documento y los podrá encontrar en la carpeta InstalarApp.

Al ejecutar el archivo de instalación, siga las instrucciones que le brinda este programa de instalación, al terminar la instalación se creará un grupo de programas donde podrá ejecutar la aplicación.

En caso de no querer ejecutar la instalación de esta aplicación, el directorio *CopiarApp*, contiene los archivos necesarios para la ejecución de esta aplicación, solo es necesario crear un directorio en su equipo y copiar todos los archivo del directorio mencionado.

Entrar a la Aplicación

Si existe un grupo de programas creado por el instalador seleccione en este grupo el elemento llamado Servicios Web para Base de Datos y automáticamente entrará a la aplicación.

En caso de no haber ejecutado el programa de instalación o en caso de querer entrar desde la línea de órdenes, el procedimiento para entrar a esta aplicación es el siguiente:

- Desde la línea de comandos ubicar el directorio de instalación.
- Ejecutar el siguiente comando:

java [**-D**DEBUG=true] **–jar** WSAdmin.jar

- Con esta operación podrá entrar a la aplicación SW4BD, el parámetro opcional DEBUG podrá contener valores TRUE o FALSE (valor por defecto), en caso de indicar un TRUE, la aplicación estará arrojando mensajes sobre el funcionamiento de la misma en el archivo WS4BD.log.

Cuando entra la aplicación mostrará el dialogo de la figura D.1, este dialogo sirve para autentificar al usuario que hará uso de la misma, este usuario solo podrá ser un usuario tipo administrador, al ser la primera vez que entre a la aplicación, el usuario deberá ser *Administrador* y la clave de acceso *admin*.

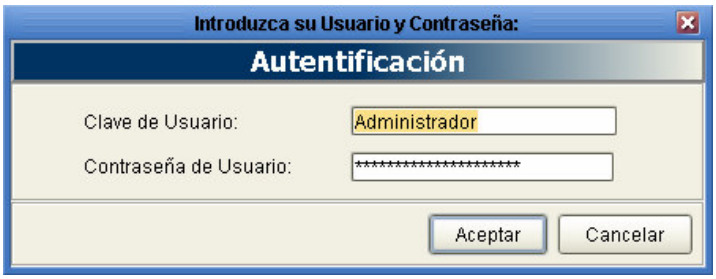


Figura D.1 Diálogo de Autentificación

Si la autentificación es correcta se mostrará la pantalla principal mostrada en la figura D.2, la cual se compone de las siguientes partes:

- **Panel de Contenido:** en la parte izquierda se muestra en forma jerárquica las opciones que son configurables para la aplicación.
- **Panel de Visualización:** en la parte central, en su caso, se muestran las opciones del elemento seleccionado en la estructura de contenido.
- **Barra de botones:** ubicada en la parte superior por debajo del menú de aplicación, contiene algunas opciones para acceder rápidamente a la configuración, además de agregar algunas opciones que no se presentan en la estructura de contenido.
- **Panel de Estado del Servidor:** ubicado en la parte inferior izquierda de la pantalla, muestra los mensajes de envío y recepción del servidor web, siempre y cuando este activo.

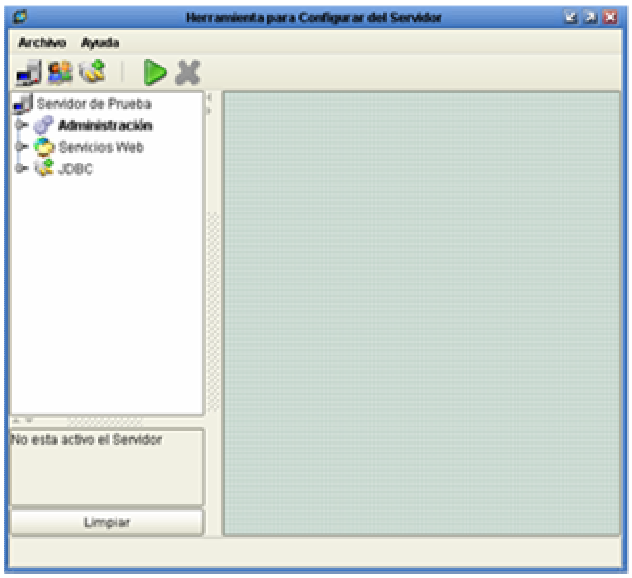


Figura D.2 Pantalla Principal.

Modificación Parámetros Servidor Web

Esta aplicación brinda un Servidor de Servicios Web para publicarlos, sin embargo la primera vez que este es utilizado deberá configurarse los parámetros del mismo, el usuario tipo administrador deberá seleccionar el primer elemento del Panel de Contenido, al seleccionarlo se mostrará la pantalla de la figura D.3, esta figura corresponde a la configuración de los parámetros del Servidor.

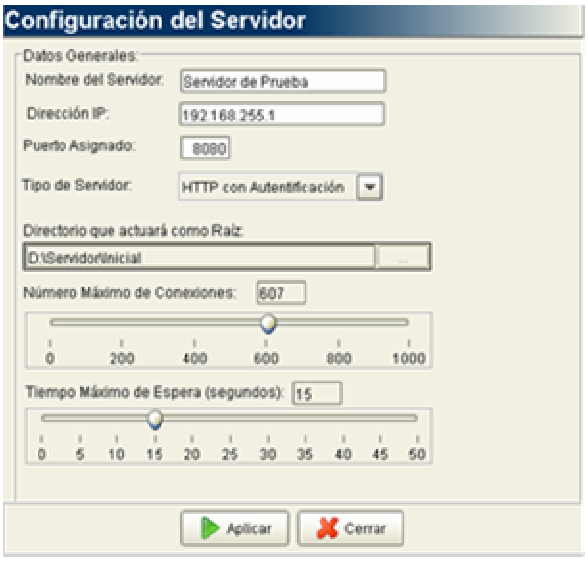


Figura D.3 Configuración de Servidor de Servicios Web.

Es importante mencionar que lo primero que debe hacerse es entrar a esta pantalla y configurar el servidor, de lo contrario no podrá realizar ninguna modificación a la aplicación.

Carga de controlador JDBC

Al estar orientado este trabajo a la publicación de Servicios Web con acceso a base de datos, es necesario contar con una opción que facilite la carga de los controladores necesarios para trabajar con la base de datos, esta función se ofrece mediante la pantalla de la figura D.4, la cual permite indicar que controladores deberá cargar la aplicación al momento de iniciar dicha aplicación, además una vez que se indica en esta pantalla los controladores a cargar y se aceptan los cambios no será necesario reiniciar la aplicación para que queden disponibles, tampoco es necesario detener el Servidor de Servicios Web, si está activo, para hacer disponible una conexión a base de datos.

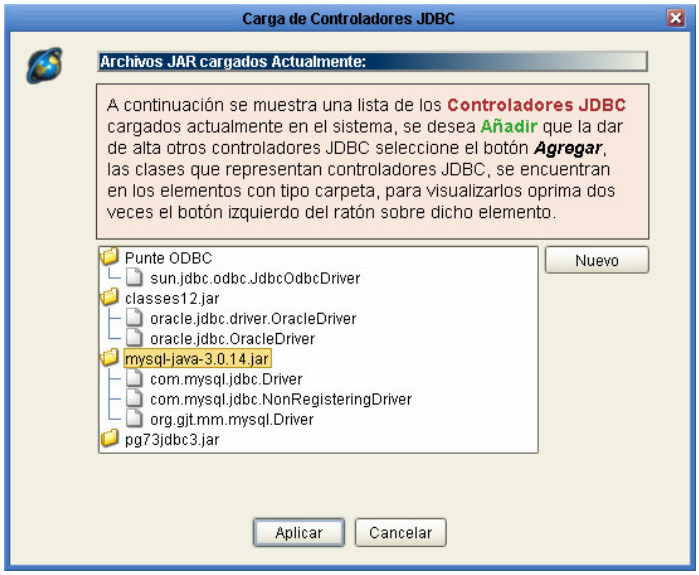


Figura D.4 Carga de Controladores JDBC.

Es importante mencionar que esta aplicación soporte los tipos de controladores 1 y 4 de JDBC, el primero permite conexiones a través de controladores ODBC, por lo que si no se cuenta con controladores JDBC tipo 4 para la fuente de datos deseada, se podrán realizar conexiones a través del puente JDBC-ODBC. El controlador tipo 1 ya viene instalado en esta aplicación.

El proceso para cargar un controlador JDBC es el siguiente:

- Abrir la pantalla de figura D.4 seleccionando el botón de la barra de herramientas.

- Seleccionar el botón *Nuevo* de esta pantalla.
- Buscar el archivo JAR necesario para el controlador JDBC y seleccionarlo, cabe aclarar que la algunos controladores JDBC necesitan más de un solo archivo por lo que será necesario cargar cada archivo a la vez.
- Seleccionar el botón *Aplicar* para aceptar los cambios realizados.

Con este proceso la aplicación pondrá disponible el controlador JDBC para poder construir conexiones a los SABD al cual pertenezca.

Conexiones JDBC

Para poder realizar Servicios Web con acceso a base de datos es necesario contar con una conexión a dicha fuente, esto solo será posible si existen controladores JDBC cargados en la aplicación tipo 4 o utilizar ODBC. La pantalla mostrada en la figura D.5, es utilizada para definir las conexiones a los SABD de acuerdo a cada controlador. En caso de no definir conexiones en esta aplicación, no podrá crear Servicios Web.

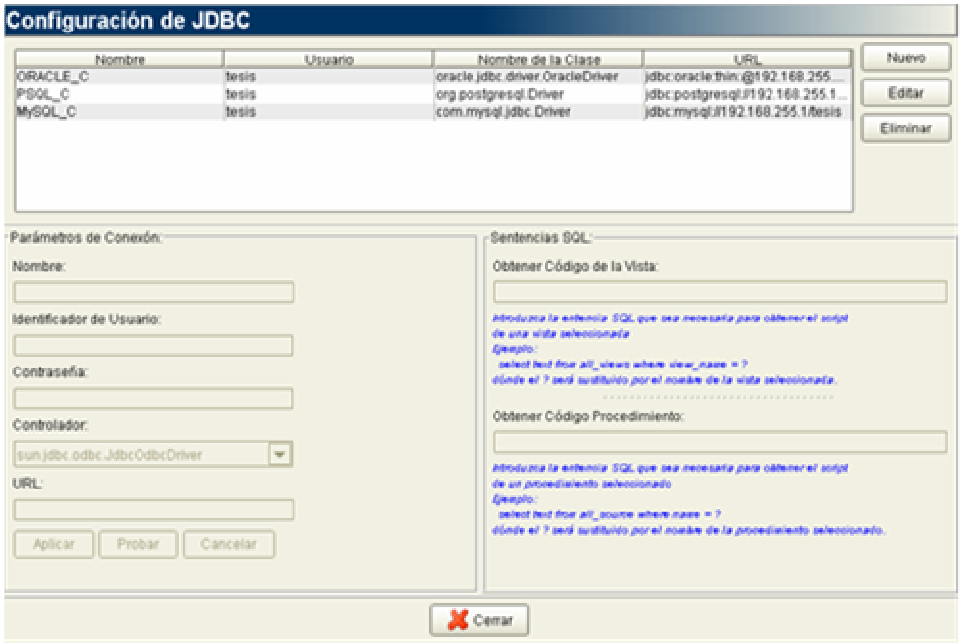


Figura D.5 Creación de conexiones JDBC.

Esta pantalla se utiliza para administrar las conexiones JDBC a los SABD, por lo que aquí puede crearse, modificarse y eliminarse las conexiones JDBC deseadas, los datos necesarios que se deben conocer para poder utilizar esta pantalla son los siguientes:

- Controlador a utilizar de acuerdo al SABD al que se desea conectar.

- Identificador y clave de acceso del usuario que establecerá la conexión.
- URL de conexión de acuerdo al controlador JDBC elegido.

Creación de Conexiones JDBC

Para crear una conexión, seleccione el botón *Agregar* de la pantalla de la figura D.5, capture los datos solicitados y oprima el botón *Probar*, en caso de que la configuración introducida sea correcta se enviará un mensaje indicando que la conexión se realizó con éxito, en caso contrario se enviará un mensaje de error.

Si la configuración es correcta, oprima el botón *Aplicar* para aceptar los cambios y crear la conexión en el archivo de configuración conf.xml.

Modificación de Conexiones JDBC

En este punto es importante tener en cuenta que la modificación de una conexión JDBC podrá afectar el comportamiento de los Servicios Web que hacen referencia a dicha conexión.

Para modificar una conexión JDBC:

- Seleccionar la conexión deseada.
- Realizar los cambios en las casillas correspondientes.
- Seleccionar el botón *Aplicar* para aceptar los cambios realizados.

Eliminación de Conexiones JDBC

Si el usuario elimina una conexión JDBC, esta aplicación le indicará que los Servicios Web que hagan referencia a dicha conexión mostrarán un mal funcionamiento, debe elegir entre continuar la eliminación ó cancelarla. En caso de mantenerlo la ejecución del Servicio Web presentará errores.

Para eliminar una conexión:

- Seleccione la conexión deseada.
- Seleccione el botón *Eliminar*.

Creación de Servicio Web

Para poder crear un Servicio Web será necesario contar ya con conexiones JDBC configuradas, en caso contrario la aplicación lo indicará y no podrá crearlos.

La creación de Servicios Web debe ser con el proceso siguiente:

- Crear un encabezado de Servicio Web.
- Crear el encabezado de las operaciones del Servicio Web.
- Crear el cuerpo para cada operación del Servicio Web.

En las secciones siguientes se detallan cada uno de los pasos a seguir.

Creación de Encabezado de Servicio Web

El encabezado del Servicio Web es el que determina como estará configurado dicho servicio, para ello el administrador deberá entrar a la pantalla de la figura D.6, seleccionando el elemento Servicios Web del Panel de Contenido.

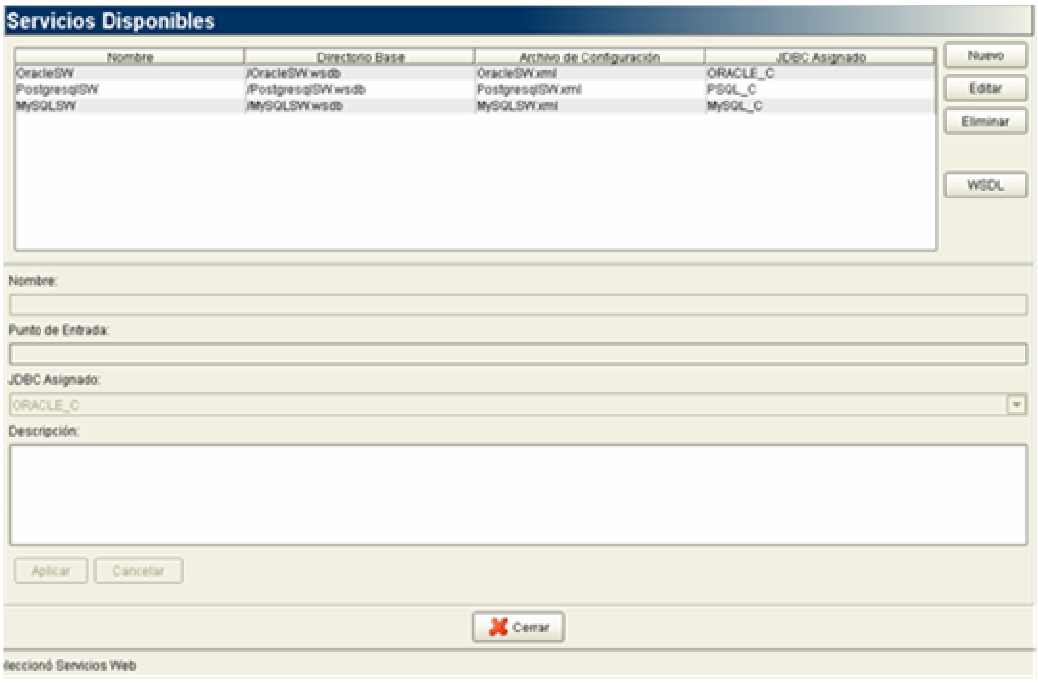


Figura D.6 Configuración de Encabezado del Servicio Web.

En la figura D.6 se observa las opciones que brinda la aplicación para cada Servicio Web, en esta pantalla el administrador puede Agregar, Modificar y Eliminar Servicios Web.

El botón *Ver WSDL*, solo funcionará para aquellos servicios que ya tengan configuradas operaciones, y sirve para visualizar en pantalla el documento que describe al Servicio Web.

Agregar

Para agregar un Servicio deberá realizar los pasos siguientes:

- Seleccionar el Botón *Nuevo*.
- Proporcionar los datos como Nombre, JDBC asignado y una descripción sobre el contenido del Servicio Web.
- Oprimir el botón *Aplicar* y aceptar el mensaje enviado por la aplicación (este mensaje es solo informativo, debido a que no se han creado operaciones).

Con estos pasos la aplicación ha creado un nuevo Servicio y aparecerá debajo del elemento Servicios en el Panel de Contenido.

Modificar

Al modificar un Servicio Web, solo podrán cambiarse los datos de JDBC asignado y la descripción. El punto crítico es el cambio de JDBC asignado, ya que si este es cambiado podrán obtenerse resultados no deseados al momento de ejecutar una operación configurada con un anterior JDBC asignado, por lo que se recomienda adaptar las operaciones ya construidas a la nueva configuración del Servicio Web.

Eliminar

Para eliminar un Servicio Web deberá seleccionarlo de la Tabla que lista los Servicios Existentes oprimir el botón *Eliminar* y aceptar o rechazar los mensajes que envía la aplicación, en caso de eliminar un Servicio Web, también se eliminarán sus operaciones y sus archivos de configuración.

Creación de Operaciones

En el Panel de Contenido, se muestran los Servicios Web que se han configurado, para poder agregarles operaciones es necesario que existen en esta parte de lo contrario no se podrán crear operaciones.

La creación de las operaciones se realiza mediante la pantalla de la figura D.7, en esta pantalla se puede observar las siguientes opciones:

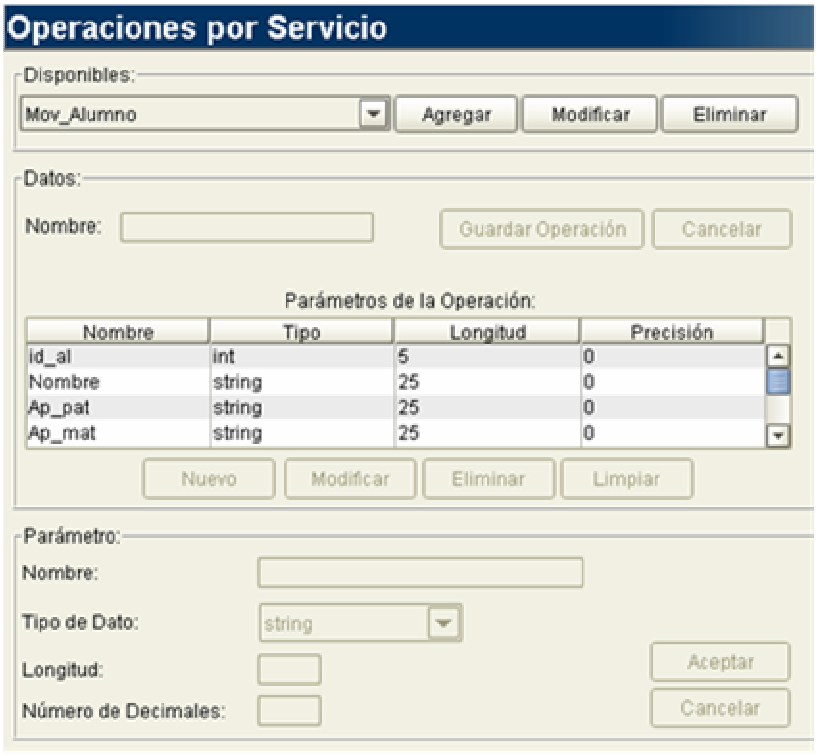


Figura D.7 Operaciones por Servicio Web.

Disponibles

Indica cuales operaciones ya se encuentran configuradas para el Servicio Web seleccionado.

Datos

Muestra la información sobre los parámetros para cada operación, pueden definirse operaciones sin parámetros.

Parámetro

Es utilizado para mostrar la información de cada uno de los parámetros de la operación y para poder agregar o eliminar parámetros, esta parte se explica mas adelante.

El botón *Limpiar* de esta sección, se encarga de quitar todos los parámetros de la operación.

Agregar

Para agregar operaciones debe realizarse el siguiente proceso:

- Seleccionar el botón *Agregar*, con esto se habilitan la parte de Datos de la operación.
- Introducir el Nombre de la operación.
- Si no cuenta con parámetros oprimir el botón *Guardar Operación* para aceptar los cambios o cancelar para rechazarlos.

Modificar

La modificación de una operación alterará tanto el documento WSDL como el comportamiento del cuerpo de dicha operación.

A una operación se podrán modificar sus parámetros pero no el nombre de la misma.

Seleccione la operación deseada y oprima el botón *Modificar*.

Eliminar

Para eliminar una operación basta con seleccionarla de la lista de las disponibles y seleccionar el botón *Eliminar*.

Parámetros de la Operación

Para asignar y modificar los parámetros de una operación, debe realizarse los siguientes pasos.

- Para agregar un nuevo parámetro seleccionar el botón *Nuevo*.
- Para modificar un parámetro, seleccionar el parámetro deseado y oprimir el botón *Modificar*.
- Para eliminar seleccionar el parámetro deseado y oprimir el botón *Eliminar*.

Si se modifica o crea un parámetro los datos a capturar dependerán del tipo de dato que se requiera, en este caso solo el tipo de datos cadena aceptará un número en la longitud. El resto de los tipos de datos será predefinido, el usuario podrá modificar el nombre, tipo y longitud (en su caso) de cada uno de los parámetros. Los parámetros aquí definidos servirán para ligarlos al cuerpo de cada operación.

Cada vez que se cree o se modifique la descripción de una operación, se regenerará automáticamente el WSDL que lo describe, el cual podrá ser localizado en el mismo directorio donde se instaló la aplicación.

Creación de Cuerpo de Operación

Para crear el cuerpo de una operación es necesario contar con la definición de la misma, como se ha mencionado este trabajo esta orientado a la publicación de Servicios Web con acceso a base de datos, por lo que le cuerpo de la misma será en base a sentencias SQL que se enviarán al SABD correspondiente para ser ejecutadas e informar al consumidor los resultados en XML.

El cuerpo de la operación se construye en la pantalla de la figura D.8, la cual esta dividida en tres partes:

- Listado de Instrucciones SQL ya configuradas, se muestra en el lado izquierdo superior.
- Tipo de instrucción a crear, localizada en la parte inferior izquierda de la pantalla.
- Y el tipo de operación a crear que se encuentra en la parte central.

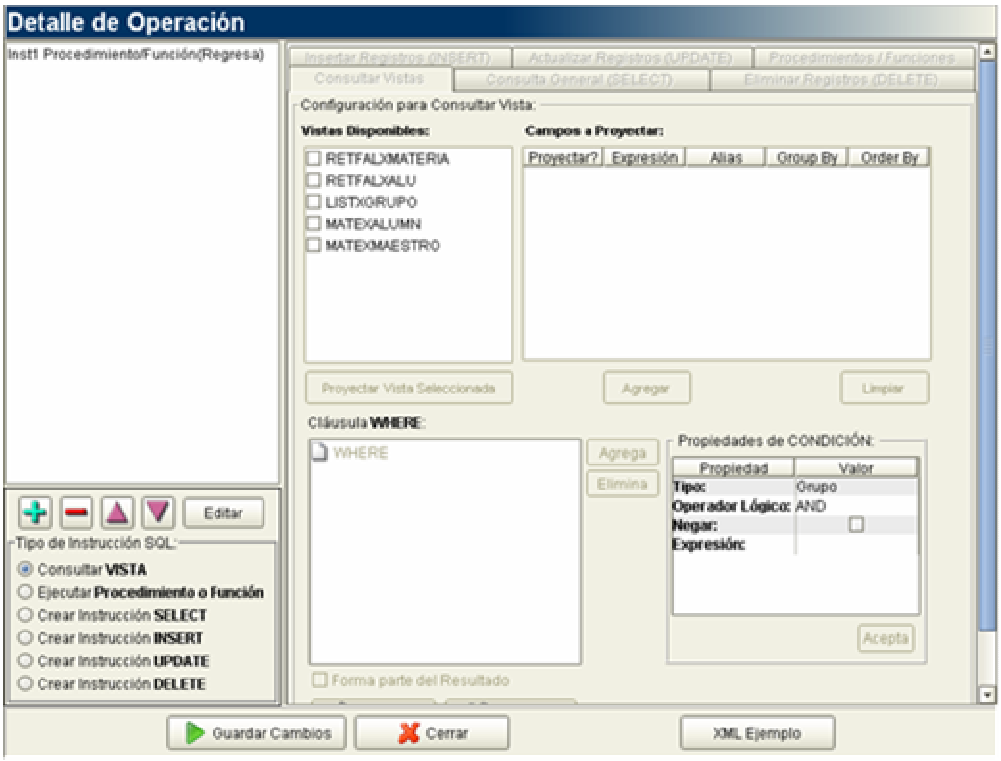


Figura D.8 Cuerpo de Operación

Para agregar instrucciones SQL es necesario seleccionar primero el tipo de instrucción que se desea construir, posteriormente seleccionar el botón  y se habilitará la pestaña correspondiente a la operación seleccionada.

El orden de la ejecución de cada instrucción puede ser alterado con los botones  de la parte inferior a la lista de las instrucciones.

Por otro lado se puede modificar una instrucción seleccionándola y oprimiendo el botón *Editar*.

Para eliminar una instrucción deberá seleccionarla y oprimir el botón .

En secciones siguientes se explica el funcionamiento de cada una de las pestañas de acuerdo al tipo de instrucción a crear y/o modificar.

El botón *XML Ejemplo*, muestra una estructura semejante de XML al que será enviado al consumidor.

Vistas / Sentencia SELECT

La mayoría de los SABD soportan guardar consultas realizadas con la cláusula SELECT como vistas para posteriormente realizar consultas sobre ellas, en esta parte de la aplicación se muestran, si el SABD lo soporta, las vistas creadas por los usuarios.

Para el caso de la sentencia SELECT, se mostrarán las tablas y las vistas para que el administrador pueda realizar las consultas a las mismas.

Para poder proyectar una tabla y/o vista se debe realizar lo siguiente:

Seleccionar los elementos deseados a proyectar.

Seleccionar el botón *Proyectar Vista/Tabla seleccionada*.

Con esto los campos de los elementos seleccionados se listarán en la tabla que se encuentra a la derecha de la lista de elementos, la tabla contiene las siguientes columnas:

- **Proyectar?:** esta columna le indica a la aplicación si el campo de ese renglón se tomará en cuenta al momento de construir la sentencia SQL a enviar al SABD.
- **Expresión:** indica que campo o expresión se tomará en cuenta en la cláusula SELECT, en esta columna se puede dejar el nombre del campo o construir una expresión más compleja seleccionando el botón **...** que aparece en el campo de la tabla, aparecerá la pantalla de la figura D.9.

  En esta pantalla se puede construir una expresión más compleja con las funciones que ofrece el SABD elegido y combinación de los campos.
- **Alias:** determina el nombre con el que será conocido el campo y/o expresión al momento de generar el resultado en XML.
- **Group By:** indica si este campo será parte o no para agrupar la consulta seleccionada.
- **Order By:** determina si el campo seleccionado se incluirá para ordenar los resultados arrojados.

Figura D.9 Construcción de Expresiones

La parte inferior de la pantalla, muestra un elemento utilizado para definir la parte de condiciones que debe cumplir una sentencia SELECT (la cláusula WHERE), para agregar condiciones se debe seleccionar el elemento del árbol en donde se desea agregar la condición o grupo, la aplicación por defecto agrega grupos por lo que en el panel derecho de esa pantalla se podrá modificar el tipo de elemento (condición o grupo), el tipo de operador que se formará (AND u OR) y si va a ser negada o no la condición construida, en la parte de expresión se podrá realizar la condición de comparación para este elemento, la pantalla mostrada es la misma de la figura D.9, pero con la lista de los parámetros de la operación habilitados, esto es para poder construir y arrojar resultados del SABD a través de los parámetros enviados por los consumidores de los Servicios Web. Deberá oprimir el botón *Aceptar* para que los cambios realizados se muestren en el elemento seleccionado de lo contrario no se realizarán.

El último elemento de esta pantalla es la casilla de verificación sobre si el resultado de la ejecución de esta instrucción SQL será informado o no al consumidor a través del XML generado.

Para aceptar los cambios realizados a esta instrucción se debe oprimir el botón *Aplicar* o en su caso *Regresar* para no aceptar los cambios.

Sentencia INSERT / Sentencia UPDATE

Esta pestaña tiene un comportamiento similar a la de Sentencia SELECT, sin embargo solo se muestran elementos tipo tabla en las que si se pueden realizar operaciones de inserción, además se presentan otro tipo de columnas en la tabla de los campos proyectados, con las que se podrán construir las expresiones que serán almacenadas en el campo elegido. La figura D.10 muestra la pantalla para la sentencia INSERT.

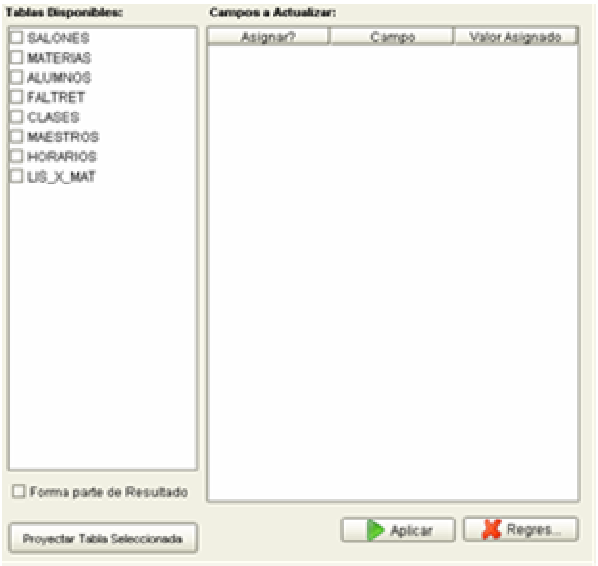


Figura D.10 Construcción de Sentencia INSERT

Para la sentencia INSERT no se presenta el panel inferior para construir condiciones, sin embargo para la sentencia UPDATE si se debe ofrecer la posibilidad de construir condiciones, por lo que en ella si se contempla esta opción, para la sentencia UPDATE la pantalla es la de la figura D.11.

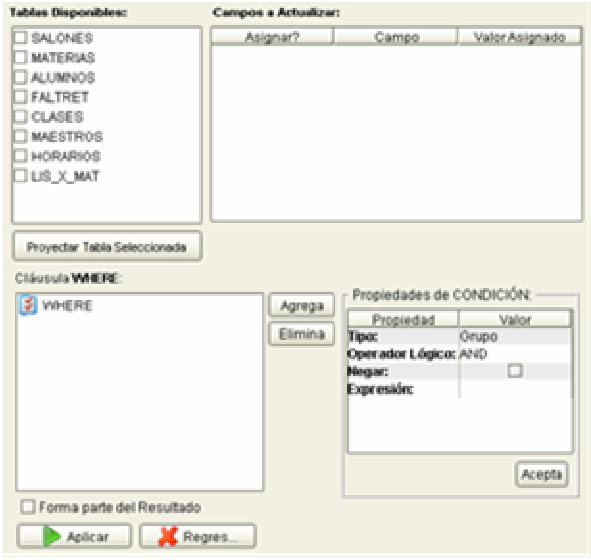


Figura D.11 Construcción de Sentencia UPDATE.

Para aceptar los cambios realizados a esta instrucción se debe oprimir el botón *Aplicar* o en su caso *Regresar* para no aceptar los cambios.

Sentencia DELETE

Esta sentencia es utilizada para eliminar registros de una tabla en base a ciertas condiciones, por lo que solo se ofrece la posibilidad de seleccionar la tabla deseada y la construcción de las condiciones.

La figura D.12, presenta la pantalla para la sentencia DELETE, el uso de la misma es similar a la pantalla de la sentencia UPDATE con la diferencia de que no existen campos a seleccionar.

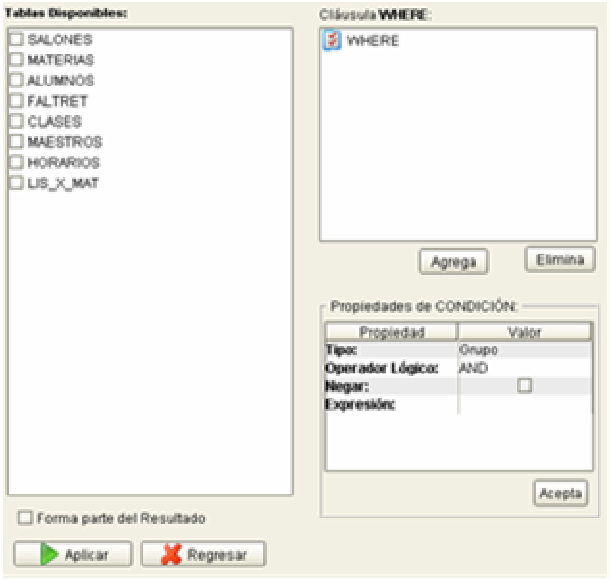


Figura D.12 Construcción sentencia DELETE.

Para aceptar los cambios realizados a esta instrucción se debe oprimir el botón *Aplicar* o en su caso *Regresar* para no aceptar los cambios.

Procedimientos y Funciones

Esta pantalla, mostrada en la figura D.13, representa la creación de instrucciones SQL que le indiquen al SABD la ejecución de algún procedimiento y/o función en el SABD.

El administrador deberá seleccionar el elemento deseado a ejecutar, este elemento contendrá ciertos parámetros que solo los expresados en la tabla de la derecha, estos parámetros corresponden al del elemento seleccionado para relacionarlos deberá crear una expresión en base a valores directos o a la asignación de los parámetros de la operación del Servicio Web. La columna Valor Asignado es la encargada de definir la construcción del parámetro a enviar al SABD.

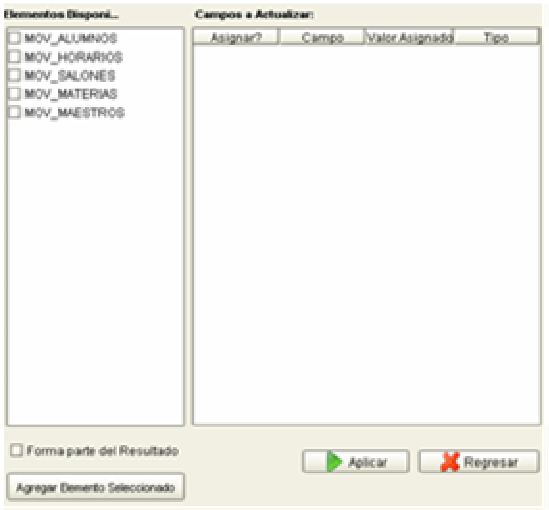


Figura D.13 Construcción de sentencia para Procedimientos y/o funciones.

Acceso a base de datos

Este es un módulo extra que el administrador puede utilizar para conectarse a la base de datos y trabajar directamente con ella a través del controlador JDBC elegido, en el se podrán ejecutar sentencias SQL directamente al SABD, además de consultar los elementos que contiene la base de datos. La figura D.14, presenta la pantalla para este módulo.

Figura D.14 Módulo de Administración SABD.

Las partes que la componen son las siguientes:

Editor SQL

El editor SQL se encarga de enviar la sentencia SQL que el administrador capture en la parte superior, en la parte inferior presenta los resultados arrojados por la ejecución, en caso de ser una sentencia que modifica los datos, informará sobre le número de registros afectados, en caso contrario presentará una tabla con los datos arrojados por dicha sentencia.

En caso de ocurrir algún error en la ejecución de la sentencia, se le indicará al usuario sobre dicho error.

Tablas

En esta pestaña se muestra un listado sobre las tablas de la base de datos a la que se encuentra conectada, al seleccionar una tabla y oprimir dos veces el botón izquierdo del ratón, se desplegarán los campos que la conforman, esta aplicación no permite crear nuevas tablas a partir de esta pestaña, si el usuario desea construir nuevas tablas deberá hacerlo a través del Editor de SQL.

Vistas

En esta pestaña se muestra un listado sobre las vistas de la base de datos a la que se encuentra conectada, al seleccionar una vista y oprimir dos veces el botón izquierdo del ratón, se desplegarán los campos que la conforman, así como la forma en que esta construida, este último dato dependerá de que si el administrador capturó la sentencia para recuperar el código de una vista en la configuración de las conexiones JDBC, en caso de que no lo capture, no se presentará la composición de la vista.

En esta pantalla se podrán crear nuevas vistas seleccionando el botón *Crear*, de la parte inferior derecha de la pantalla, con esto se habilita el área de texto donde se puede introducir una sentencia SQL para crear la vista. La sentencia SQL dependerá del SABD al que se conecta esta conexión JDBC.

Procedimientos

En esta pestaña se muestra un listado sobre los procedimientos y/o funciones de la base de datos a la que se encuentra conectada, al seleccionar uno de estos elementos se mostrará su información en la parte derecha de la pantalla, tanto la información de los parámetros del mismo, como el código con la que fue construido, este último dato dependerá de que si el administrador capturó la sentencia para recuperar el código de una procedimiento y/o función en la configuración de las conexiones JDBC, en caso de que no lo capture, no se presentará el código fuente de este elemento.

En esta pantalla se podrán crear nuevas funciones o procedimientos seleccionando el botón *Crear*, de la parte inferior derecha de la pantalla, al seleccionarlo se pregunta el tipo de elemento a crear si es un procedimiento y/o función, después de elegir el tipo de elemento y si la aplicación cuenta con la información suficiente para esta conexión se mostrará una plantilla para la creación de dicho elemento.

La codificación del elemento dependerá del SABD y del lenguaje que soporte para su construcción.

Servidor Web

Esta aplicación cuenta con un servidor para publicar los Servicios Web configurados, este Servidor está orientado sólo a la publicación de los mismos por los que le tipo de solicitudes que acepta son:

- **Solicitud de Archivo WSDL:** el consumidor está solicitando la descripción de algún Servicio Web.
- **Recepción de mensajes SOAP:** cuando el consumidor solicita la ejecución de alguna operación del Servicio Web.

Este servidor esta construido para escuchar peticiones HTTP, por lo que podrá revisar si está funcionando introduciendo la siguiente URL desde su explorador de Internet:

http://localhost:[puerto]

donde *puerto* es el número de puerto configurado par escuchar las peticiones de los consumidores.

Iniciar

Para iniciar el Servidor de Servicios Web simplemente oprima el botón  de la barra de herramientas, con esto la aplicación comienza el proceso siguiente:

- Verifica que no se estén cambiando las propiedades del Servidor.
- Revisa las conexiones JDBC disponibles y las activa.
- Inicia el proceso para escuchar peticiones en el puerto seleccionado y atender las mismas.

Si en la ejecución de este proceso no ocurre ningún error, el servidor estará listo par escuchar las peticiones de los consumidores.

Cuando el servidor esta activo, el administrador puede seguir construyendo Servicios Web, la única operación que no puede realizar es el cambio de la configuración del Servidor.

El servidor estará enviando mensajes en el Panel de Estado del Servidor indicando las peticio0nes que ha recibido y la respuesta que se ha enviado.

Detener

Para detener el Servidor de Servicios Web sólo es necesario oprimir el botón  de la barra de herramientas con esto se realizará el proceso siguiente:

- Termina de atender las solicitudes activas.
- Cierra el proceso que escucha peticiones.
- Cierra las conexiones JDBC que se encuentren activas para la atención de Servicios Web.

Después de este proceso el Servidor estará inactivo y se indicará con un mensaje en el Panel de Estado del Servidor.

Archivo LOG

Cuando se ejecuta por primera vez esta aplicación, se genera un archivo llamado *ws4bd.log*, el cual contiene información útil para saber el comportamiento de la aplicación, este archivo crecerá a medida que se utilice la aplicación, en caso de eliminarlo, se crea un nuevo archivo vacío con el mismo nombre.

La información arrojada en el archivo log, contendrá mensajes de error y mensajes propios de la aplicación indicando el tipo de mensaje, la hora que ocurrió y el contenido del mensaje.

Si el administrador ha ejecutado la aplicación con la opción **–DDEBUG=true** entonces en este archivo se mostrará información mas detallada sobre la operación de la aplicación.

Anexo E
Contenido del CD-ROM

El disco compacto que acompaña a este documento se encuentra organizado de la siguiente manera:

Directorio	Descripción
SW4BD	Directorio Principal de la aplicación
Instalar	Directorio que tiene los diferentes archivos o directorios para llevar a cabo la instalación de esta aplicación
InstalarApp	Contiene los archivos binarios para poder ejecutar un instalador en base a la plataforma deseada.
CopiarApp	Contiene los archivos iniciales necesarios para ejecutar esta aplicación, por lo que este directorio puede ser copiado a la máquina donde se instalará esta aplicación y poder trabajar con ella.
CodigoFuente	Presenta el código fuente utilizado para la generación de la aplicación.
Documentacion	Contiene este documento.
JAVA	Contiene algunas librerías utilizadas para el desarrollo de la aplicación y algunos controladores JDBC para poder ser utilizados en la misma.
JavaJRE	Contiene los archivos que de acuerdo a la plataforma deberán ejecutarse en caso de no contar con el JRE de Java.
JDBC	Contiene algunas librerías con controladores JDBC tipo 4 para algunos SABD existentes en el mercado
SABD	Contiene archivos relacionados con los SABD y las bases de datos.
SQL	Contiene los script para la creación de la base de datos ejemplo para MySQL, PostgreSQL y Oracle.
MySQL	Contiene la versión 4.x de MySQL para poder probar esta aplicación.
XML	Contiene clases Java que se encargan de interactuar con los archivos de configuración, así como el manejo de mensajes SOAP y la generación de los archivos WSDL de cada Servicio Web.

Anexo F
Código Fuente

Para consultar el código fuente de este trabajo, por favor consulte el directorio correspondiente de acuerdo a los listados en el Apéndice E.

Referencias

Bibliográficas

[1] Harvey Deitel, Paul Deitel, B. DuWaldt, L. Trees Agosto 2002, WEB SERVICES A TECHNICAL INTRODUCTION Editorial Prentice Hall Ptr.

[2] David A, Chappell & Tyler Jewell Marzo 2002, JAVA WEB SERVICES Editorial O'Reilly

[3] Robert Englander Mayo 2002, JAVA AND SOAP Editorial O'Reilly.

[4] James Snell, Doug Tidwell, Pavel Kulchenko Diciembre 2001, PROGRAMMING WEB SERVICES WITH SOAP Editorial O'Reilly.

[5] Henry Bequet Diciembre 2001, PROFESSIONAL JAVA SOAP Editorial Wrox Press Inc.

[6] William B. Brogden, Bill Brogden Enero 2002, SOAP PROGRAMMING WITH JAVA PUBLISHER Editorial Sybex Inc.

[7] Aaron E Walsh Abril 2002, UDDI, SOAP, AND WSDL: THE WEB SERVICES SPECIFICATION REFERENCE BOOK Editorial Prentice Hall.

[8] Scott Short Febrero 2002, BUILDING XML WEB SERVICES FOR THE MICROSOFT .NET PLATFORM Editorial Microsoft Press.

[9] Patrick Cauldwell, Rajesh Chawla, Vivek Chopra Junio 2002, SERVICIOS WEB XML Editorial Anaya Multimedia.

[10] Brian E. Travis Agosto 2000, XML AND SOAP PROGRAMMING Editorial Microsoft Press.

[11] Ethan Cerami Febrero 2002, WEB SERVICES ESSENTIALS Editorial O'Reilly.

[12] Michael Leventhal, David Lewis, Matthew Fuchs Enero 1998, DESIGNING XML INTERNET APPLICATIONS Editorial Prentice Hall PTR.

[13] Charles f. Goldfarb, Paul Prescod; traduccion: Beatriz Paredes, Alejandro Ruiz, Belen Venegas Junio 1999, MANUAL DE XML Editorial Pretince Hall.

[14] Jake Sturm Septiembre 2000, DEVELOPING XML SOLUTIONS Editorial Microsoft Press.

[15] William R. Stanek, William Stanek Enero 2000, PROFESSIONAL XML DATABASES Editorial Wrox Press.

[16] Michael Young Julio 2000, XML STEP BY STEP Editorial Microsoft Press.

[17] Elliotte Rusty Harold Septiembre 2003, XML BIBLE Editorial Hungry Minds.

[18] John Robert Gardner and Zarella L. Rendon Julio 2001, XSLT AND XPATH: A GUIDE TO XML TRANSFORMATIONS Editorial Prentice Hall.

[19] Harvey M. Deitel, Paul J. Deitel, T. R. Nieto, Ted Lin, Praveen Sadhu Diciembre 2000, XML: HOW TO PROGRAM Editorial Prentice Hall.

[20] Fabio Arciniegas A. Julio 2002, XML DEVELOPER´S GUIDE Editorial McGraw-Hill

[21] Aaron Skonnard, Martin Gudgin Octubre 2001, ESSENTIAL XML QUICK REFERENCE: A PROGRAMMER'S REFERENCE TO XML, XPATCH, XSLT, XML SCHEMA, SOAP, AND MORE Editorial Addison-Wesley.

[22] Bhavani Thuraisingham Marzo 2002, XML DATABASES AND THE SEMANTIC WEB Editorial CRC Press.

[23] Graeme Malcolm; traduccon Jose Antonio Bautista Montejo Junio 2001, PROGRAMACIÓN DE MICROSOFT SQL SERVER 2000 CON XML Editorial McGraw-Hill.

[24] Mike Jasnowski Enero 2002, JAVA XML, AND WEB SERVICES BIBLE Editorial Hungry Minds

[25] Casey Kochmer, Erica Frandsen Marzo 2002, JSP AND XML: INTEGRATING XML AND WEB SERVICES IN YOUR JSP APPLICATION Editorial Addison-Wesley.

[26] C. J. Date Octubre 1999, AN INTRODUCTION TO DATABASE SYSTEMS Editorial Addison Wesley.

[27] Alfonso F. Cardenas Enero 1985, DATA BASE MANAGEMENT SYSTEM Editorial William C Brown.

[28] Kyle Geiger; with programming by Brian Tschumper and Jason Zander Agosto 1995, INSIDE ODBC Editorial Microsoft Press.

[29] David Sussman Agosto 2000, ADO 2.6 PROGRAMMER'S REFERENCE Editorial Wrox Press.

[30] Robert Signore, John Creamer, Michael O. Stegman Febrero 1995, THE ODBC SOLUTION: OPEN DATABASE CONNECTIVITY INDISTRIBUTED ENVIRONMENTS Editorial McGraw-Hill.

[31] Dan D. Gutierrez Agosto 1999, WEB DATABASE DEVELOPMENT FOR WINDOWS PLATFORMS Editorial Pretince Hall.

[32] Patrik Patel, Karl Moss Julio 1997, JAVA DATABASE PROGRAMMING WITH JDBC Editorial Coriolis Group Books.

[33] George Reese Enero 2000, DATABASE PROGRAMMING WITH JDBC AND JAVA Editorial O'Reilly.

[34] Graham Hamilton, Rick Cattell, Maydene Fisher Agosto 1997, JDBC DATABASE ACCESS WITH JAVA: A TUTORIAL AND ANNOTATED REFERENCE Editorial Addison Wesley.

[35] Van Haeke, Bernard Enero 2002, JDBC 3.0: JAVA DATABASE CONNECTIVITY Editorial Hungry Minds.

[36] Eric Jung, Andrei Cioroianu, Dave Writz, Mohammad Akif, Steven Brodhead, James Hart Julio 2001, JAVA XML PROGRAMMER'S REFERENCE Editorial Wrox Press Inc.

[37] Elliote Rusty Harold Agosto 2000, JAVA NETWORK PROGRAMMING, 2nd Edition Editorial O'Reilly.

[38] Steven J. Gutz, Steven Gutz, Matthew Robinson, Pavel Vorobiev Septiembre 1999, UP TO SPEED WITH SWING: USER INTERFACES WITH JAVA FOUDATION CLASSES Editorial Manning Publications.

[39] Kim Topley Diciembre 1999, CORE SWING: ADVANCED PROGRAMMING Editorial Prentice Hall Ptr.

[40] Paul Hyde Agosto 1999, JAVA THREAD PROGRAMMING Editorial Sams.

[41] Scott Oaks, Henry Wong Enero 1999, JAVA THREADS Editorial O'Reilly.

[42] Harvey M. Deitel, Paul J. Deitel Agosto 2001, JAVA: HOW TO PROGRAM Editorial Prentice Hall.

[43] Doug Lea Noviembre 1999, CONCURRENT PROGRAMMING IN JAVA: DESIGN PRINCIPLES AND PATTERNS Editorial Addison-Wesley

[44] Marco Pistoia, Duane F. Reller, Deepak Gupta, Milind Nagnur, Ashok K. Ramani, Marco Pistoia, Ashok Ramani Agosto 1999, JAVA NETWORK SECURITY Editorial Prentice Hall Ptr.

[45] Jeff Magee and Jeff Kramer Abril 1999, CONCURRENCY: STATE MODELS & JAVA PROGRAMS Editorial Wiley.

[46] Martin Rineart 1998, DESARROLLO DE BASES DE DATOS EN JAVA Editorial McGraw-Hill.

[47] Elliote Rusty Harold Marzo 1999, JAVA I-O Editorial O'Reilly.

[48] Dick Steflik, Prashant Sridharan, Richard Steflik Abril 2000, ADVANCED JAVA NETWORKING Editorial Prentice Hall Ptr.

[49] JoAnn Hackos, Dawn Stevens STANDARS FOR ONLINE COMMUNICATION: PUBLISHING INFORMATION FOR THE INTERNET/WOLD WIDE WEB, HELP SYSTEMS/CORPORATE INTRANET Editorial New York: John Wiley.

[50] Clinton Wong, Lincoln Stein, Ron Petrusha, Shishir Gundavaram Abril 1997, SCRIPTING LANGUAGES: AUTOMATING THE WEB Editorial O´ Reilly.

Electrónicas

[L-1] Sistemas Administradores de Bases de Datos, http://www.unalmed.edu.co/~mstabare/Dbms.htm.

[L-2] Grupo de Investigación en bases de datos, http://dis.unal.edu.co/grupos/unbd/index.html.

[L-3] WEB DATABASE INTEGRATION, http://www.ravens-nest.com/support/papers/Web-DatabaseDesignReport3.pdf.

[L-4] Tutoriales soaprpc.com, http://www.soaprpc.com/tutorials/.

[L-5] Using WSDL in SOAP applications, http://www-106.ibm.com/developerworks/webservices/library/ws-soap/index.html?dwzone=ws.

[L-6] SOAP, the Simple Object Access Protocol, http://www.wdvl.com/Authoring/Languages/XML/Soap/index.html.

[L-7] Technology Reports: Simple Object Access Protocol (SOAP), http://www.oasis-open.org/cover/soap.html.

[L-8] Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/REC-xml/.

[L-9] WebServices – Axis, http://ws.apache.org/axis/.

[L-10] W3Schools - Full Web Building Tutorials, http://www.w3schools.com.

[L-11] Web Service Definition Language (WSDL), http://www.w3.org/TR/wsdl.

[L-12] SOAP Specifications, www.w3.org/TR/soap/.

[L-13] An Overview of the SOAP v1.1 Specification, http://www.developer.com/services/article.php/641361.

[L-14] Hypertext Transfer Protocol, http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[L-15] O'Reilly XML, http://www.xml.com/.

[L-16] O'Reilly WebServices, http://webservices.xml.com/.

[L-17] O'Reilly Network, http://www.oreillynet.com/.

[L-18] Web Services Architect, http://www.webservicesarchitect.com.

[L-19] Servicios Web MC Carlos Lizárraga Celaya, Laboratorio de Física Interdisciplinaria, Departamento de Física Universidad de Sonora http://www.fisica.uson.mx/carlos/WebServices/WSRevolution.htm.

[L-20] Dynamically Integrating XML Web Services With Relational Databases, http://www.mywebservices.org/index.php/article/view/912/.

[L-21] WebServices.org, http://www.webservices.org/.

[L-22] SOAP and Web Services, Stanford University, http://www.mnot.net/papers/ws-stan.html.

[L-23] GotDotNet: The Microsoft .NET Framework Community, http://www.gotdotnet.com/.

[L-24] TheServerSide.com, Your Enterprise Java Community, http://www.theserverside.com.

[L-25] Microsoft .NET, http://www.microsoft.com/net/technical/.

[L-26] Java Web Services Development Kit, http://java.sun.com/webservices/jwsdp/index.jsp.

[L-27] IBM WebSphere, http://www-306.ibm.com/software/info1/websphere/index.jsp.

[L-28] Apache SOAP, http://ws.apache.org/soap/.

[L-29] WebServices – Axis, http://ws.apache.org/axis/.

[L-30] Oracle JDeveloper, http://www.oracle.com/technology/pub/articles/index.html.

[L-31] Virtuoso Universal Server, http://www.openlinksw.com/virtuoso/Whitepapers/index.htm.

[L-32] Comercio Electronico en Mexico, http://www.e-global.es/article73.html.

[L-33] iODBC.org, http://www.iodbc.org/.

[L-34] ODBC and the unixODBC Project, http://www.unixodbc.org/.

[L-35] Evans Data Corporation, http://www.evansdata.com/survey_webservices_topical.shtml.

[L-36] XML: The Future of The World Wide Web, Ekta Agarwal, http://www.ittc.ku.edu/~sgauch/767/XML.doc.

[L-37] XML in 10 points by Bert Bos: http://www.w3.org/XML/1999/XML-in-10-points, traducción Gustavo García: http://www.w3.org/XML/1999/XML-in-10-points.es.html.

[L-38] HTTP - Hypertext Transfer Protocol, http://www.w3.org/Protocols/.

[L-39] JDBC Specification, http://java.sun.com/products/jdbc/download.html.

[L-40] Altova XML Spy, http://www.altova.com/products_ide.html.

[L-41] The Impact of XML on Databases and Data Sharing, Len Seligman and Arnon Rosenthal, http://www.mitre.org/work/tech_papers/tech_papers_00/seligman_impact/seligman.pdf.

[L-42] XML and Data Integration, Elisa Bertino and Elena Ferrari, http://www.sei.cmu.edu/isis/references/technology/xml-data.pdf.

[L-43] Building Reliable Web Services Compositions, Paulo F. Pires, Mário R. F. Benevides, and Marta Mattoso, http://www.netobjectdays.org/pdf/02/papers/ws-rsd/1251.pdf.

[L-44] Web Services: Why and How, Francisco Curbera, William A. Nagy and Sanjiva Weerawarana IBM T.J. Watson Research Center, http://www.research.ibm.com/people/b/bth/OOWS2001/nagy.pdf.

[L-45] Interoperability of Multiple Autonomous Databases, W. Litwin, L. Mark and N. Roussopoulos, http://techreports.isr.umd.edu/reports/1989/TR_89-12.pdf.

[L-46] The Network as a Global Database: Challenges of Interoperability, Proactivity, Interactiveness and Legacy, Peter C. Lockemann, Ulrike Kijlsch, Arne Koschel, Ralf Kramer, Ralf Nikolai, Mechtild Wallrath, Hans-Dirk Walter, http://www.vldb.org/conf/1997/P567.PDF.

[L-47] Database Integration: The Key to Data Interoperability, Christine Parent and Stefano Spaccapietra, http://lbdsun.epfl.ch/e/publications/articles.pdf/OObook.pdf.

[L-48] Levels of Information Systems Interoperability (LISI), United States Departament of Defense, http://www.defenselink.mil/nii/org/cio/i3/lisirpt.pdf.

[L-49] A Survey on Information Systems Interoperability, Renato Fileto, Claudia Bauzer Medeiros, www.dcc.unicamp.brzSzic-tr-ftpzSz2003zSz03-30.pdf/fileto03survey.pdf.

[L-50] Towards Interoperability in Heterogeneous Database Systems, A. Zisman J. Kramer, http://www.doc.ic.ac.uk/research/technicalreports/1995/DTR95-11.pdf.

Glosario

ADO: (Access Data Object, Objeto de acceso a datos), interfaz de acceso a datos usado para comunicar OLEDB Data Sources, como MS-SQL Server. Es una Interfaz a nivel aplicación que usa OLEDB, una librería de Objetos COM que permite el acceso a diversas fuentes de datos.

Administrador de Base de datos: individuo responsable del mantenimiento físico y lógico de la base de datos.

API: conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos API comunes a los que deben ajustarse todos los desarrolladores de aplicaciones.

Aplicación: software que realiza una función particular para el usuario.

Archivo: área denominada en un dispositivo de almacenamiento secundario que contiene un programa, datos o material textual.

Archivo WSDL: archivo utilizado para describir Servicios Web y que permite a las aplicaciones el describirle a otras aplicaciones, las reglas para interactuar entre si.

Argumento: parte de una función que identifica los datos sobre los cuales se debe operar.

Autentificación: firma electrónica. Tecnología que garantiza que una transmisión electrónica procede del origen que la emite.

Biblioteca de funciones: conjunto de rutinas de programa.

Base de Datos: es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior.

Browser: Visor o examinador. Programa cliente y herramienta básica de navegación para buscar los diferentes recursos de Internet. Los más usados son Netscape Navigator, Microsoft Internet Explorer, y Mosaic de la NCSA.

Bytecode: cuando en una computadora se compila un programa escrito en Java, lo que se genera es un bytecode, o archivo binario cuyo contenido, a diferencia de un archivo ejecutable, no puede ser interpretado directamente por la computadora, sino que necesita ser ejecutado en una máquina virtual de Java, que es un programa que interpreta el bytecode y transmite las instrucciones a la computadora.

Canal (Medio) de Comunicaciones: la instalación por la que se transmiten datos entre localidades de una red de computación.

CGI: abreviatura de interfaz común de puerta de enlace, software que facilita la comunicación entre un servidor Web y los programas que funcionan fuera del servidor, por ejemplo, los programas que procesan formularios interactivos o los que buscan en las bases de datos del servidor la información solicitada por un usuario.

Clase: en programación orientada a objetos, un tipo de datos definido por el usuario que especifica un conjunto de objetos que comparten las mismas características. Un miembro de la clase (objeto) es un "ejemplo" o caso de la clase. Las clases concretas están diseñadas para citar como ejemplos, mientras que las clases abstractas, para pasar las características por herencia.

Cliente/Servidor: modelo lógico de una forma de proceso cooperativo, independiente de plataformas hardware y sistemas operativos. El concepto se refiere más a una filosofía que a un conjunto determinado de productos. Generalmente, el modelo se refiere a un puesto de trabajo o cliente que accede mediante una combinación de hardware y software a los recursos situados en una computadora denominado servidor.

Código de Acceso: combinación de letras, números y signos que debe introducirse para obtener acceso a un programa o partes de un programa determinado, una terminal ó computadora personal, un punto en la red, etc.

Código Fuente: programa en su forma original, tal y como fue escrito por el programador, el código fuente no es ejecutable directamente por el computador, debe convertirse en lenguaje de maquina mediante compiladores, ensambladores o interpretes.

Comando: instrucción dirigida a una computadora que invoca la ejecución de una secuencia de instrucciones programada previamente.

Componente: una unidad de composición de aplicaciones que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio.

Conectividad: referente al grado en que los dispositivos de hardware, el software, y las bases de datos se pueden relacionar funcionalmente entre sí.

Controlador: software o pequeños programas necesarios para aprovechar al máximo los dispositivos, accesorios o periféricos distintos que se le pueden conectar a una computadora. Por otro lado también es utilizado para poder lograr conectividad con diversas aplicaciones, principalmente base de datos.

Conexión: circuito virtual de transporte que se establece entre dos programas de aplicación con fines comunicativos.

Contraseña: combinación de letras, números y signos que debe introducirse para obtener acceso a un programa o partes de un programa determinado, una terminal ó computadora personal, un punto en la red, etc.

CSS: (Hojas de Estilo) lenguaje formal de computadora usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

Dato: Se le considera la unidad mínima y básica de toda información; es un elemento susceptible de observación directa ya que no dice algo por si mismo, es decir, los datos no

son significativos como tales si no son procesados y convertidos para integrarse a toda la información.

Dirección IP: las siglas IP significan Internet Protocol, es decir Protocolo de comunicaciones para Internet. Representación o notación de la dirección de un equipo conectado a Internet o a una red con el protocolo TCP/IP.

Dirección URL: abreviatura de localizador uniforme de recursos. Es la dirección que especifica la ubicación electrónica de un sitio (site) o un recurso (un archivo) específico de Internet. Una dirección URL consta normalmente de cuatro partes: protocolo, servidor (o dominio), ruta de acceso y nombre de archivo.

Firewall: un cortafuegos (del inglés, Firewall) es un equipo de hardware o software utilizado en las redes de computadoras para prevenir algunos tipos de comunicaciones prohibidos por la política de red.

HTML: lenguaje de programación de páginas Web. Se escribe utilizando identificadores. Estos son órdenes que se dan al navegador para que muestre una página Web.

HTTP: abreviatura de Hipertext Transmission Protocol, en español: Protocolo de transferencia de HiperTexto. Es el protocolo en que se basa la tecnología de World Wide Web, consistente en un conjunto de reglas que gobiernan el software que transporta los documentos HTML a través de Internet.

Información: es la emisión o procesamiento de los datos, lo que puede proporcionar un conocimiento o la compresión de ciertos factores.

Interface: en programación orientada a objetos, permiten acceder a métodos con el mismo nombre en diferentes clases.

Interfaz de usuario: Engloba la forma en la que el operador interactúa con la computadora, los mensajes que éste recibe en pantalla, las respuestas de la computadora a la utilización de periféricos de entrada de datos, etc.

Internet: es una red de redes a escala mundial de millones de computadoras interconectadas con el conjunto de protocolos TCP/IP. También se usa este nombre como sustantivo común y por tanto en minúsculas para designar a cualquier red de redes que use las mismas tecnologías que la Internet, independientemente de su extensión o de que sea pública o privada.

Interoperabilidad: Característica de las computadoras que permite su interconexión y funcionamiento conjunto de manera compatible. Esto no siempre es posible, debido a los diferentes sistemas operativos y arquitecturas de cada sistema, pero los esfuerzos de estandarización están permitiendo que cada vez sean más las computadoras capaces de interoperar entre sí.

Intranet: red privada dentro de una organización. Las Intranet suelen utilizar protocolos de Internet para entregar contenido. A menudo se protegen contra al acceso desde Internet mediante servidores de seguridad.

JDBC: tecnología utilizada por el Lenguaje Java para acceder a Base de Datos.

Java: plataforma de software desarrollada por Sun Microsystems. Esta plataforma ha sido desarrollada de tal manera que los programas desarrollados para ella puedan ejecutarse de la misma forma en diferentes tipos de arquitecturas y dispositivos computacionales.

Lenguaje de Programación: técnica estándar de comunicación para ordenarle instrucciones a la computadora. Un lenguaje le da la capacidad al programador de especificarle a la computadora, qué tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano.

Máquina Virtual: capacidades de procesamiento de un sistema de computación creado por medio de software (en ocasiones mediante hardware) en un sistema de computación distinto.

Middleware: término común que se aplica al software que intercambia información en forma transparente entre aplicaciones y bases de datos. El Middleware es un mecanismo de conexión abstracta entre el software de aplicación y la base de datos, y oculta al programador de aplicaciones los elementos específicos que dependen de la implementación.

Multiplataforma: término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

ODBC: estándar de acceso a bases de datos desarrollado por Microsoft Corporation.

OLE DB: parte central de la tecnología de conectividad de base de datos de Microsoft.

Página Web: documento el cual esta basado en el lenguaje HTML, la cual permite obtener información de cualquier tipo.

Paquete: uno o más programas diseñados para realizar una tarea de procesamiento particular.

PDF: (del inglés Portable Document Format, Formato de Documento Portátil) es una forma de almacenamiento de documentos, desarrollado por la empresa Adobe, que es independiente de la plataforma y puede ser enviado a través de un medio de comunicación como es Internet sin perder su configuración original.

Plataforma: se define como el funcionamiento, estructura y diseño de computadoras. Dentro de este término se engloban aspectos como formato de instrucción, modo de direccionamiento y conjunto de instrucciones.

Procedimiento Almacenado: son instrucciones de SQL precompiladas, lo cual ayuda a mejorar el desempeño al momento de utilizarlos.

Protocolo: sistema de reglas o estándares para comunicarse a través de una red, en especial a través de Internet. Los equipos y las redes interactúan de acuerdo con los protocolos que determinan el comportamiento que cada lado espera del otro en la transferencia de información.

Proxy: computadora, servidor o Software que actúa como intermediario entre una computadora y la red Internet. Puede ofrecer algunos servicios adicionales de seguridad, administración y caché.

Red de Área Local (LAN): Conexión entre computadoras mediante un medio de transmisión dentro de una distancia que no supero los 10 kilómetros. Son utilizadas principalmente en edificios donde se encuentran oficinas, universidades, hospitales y centros de investigación (entre otros).

Red de Área Amplia (WAN): interconexión de computadoras en áreas geográficas distantes; dicha interconexión necesita contar con mecanismos de acceso especiales, los cuales se enmarcan en lo que se conoce como intercomunicación remota, requiriendo para ello el empleo de microondas, fibra óptica, cable submarino y satélites.

SABD: controla la organización, almacenamiento, recuperación, seguridad e integridad de los datos en una base de datos. Acepta solicitudes de la aplicación y ordena al sistema operativo transferir los datos apropiados.

Script: secuencia de instrucciones que en ocasiones puede llegar a ser una aplicación, que va "Pegado" a un programa o archivo que se ejecuta a la vez.

Servicios Web: componentes de 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.

Servidor Web: programa que implementa el protocolo HTTP. Se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP y de acuerdo a las mismas enviar un resultado.

Sistema Operativo: conjunto de programas o software para permitir comunicarse al usuario con una computadora y gestionar sus recursos de manera cómoda y eficiente. Comienza a trabajar cuando se enciende la computadora, y gestiona el hardware de la máquina desde los niveles más básicos.

Sitio Web: colección de páginas Web relacionadas, que residen en el mismo servidor y están interconectadas entre sí mediante vínculos de hipertextos que facilitan el desplazarse ordenadamente a través de las páginas que lo forman.

Sitios Dinámicos: un sitio Web que se genera de forma dinámica en base a las solicitudes enviadas por el usuario.

SOAP: estándar creado por World Wide Web Consortium que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Software: parte lógica de la computadora, esto es, el conjunto de instrucciones (programas) que puede ejecutar el hardware para la realización de las tareas de computación a las que se destina. Es por tanto un campo de estudio de la informática.

SQL: lenguaje declarativo de acceso a Bases de Datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

UML: lenguaje para el modelado unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los componentes de un Sistema.

Usuario: cualquier individuo que interactúa con la computadora a nivel de aplicación. Los programadores, operadores y otro personal técnico no son considerados usuarios cuando trabajan con la computadora a nivel profesional.

Vista: compuesta por una instrucción SQL de consulta (SELECT), que puede ser sencilla o compleja, la cual se encuentra almacenada y precompilada dentro del SABD.

WWW: (del inglés, Telaraña Mundial), la Web o WWW, es un sistema de hipertexto que funciona sobre Internet. Para ver la información se utiliza una aplicación llamada navegador Web para extraer elementos de información (llamados "documentos" o "páginas Web") de los servidores Web (o "sitios") y mostrarlos en la pantalla del usuario.

XHTML: lenguaje de marcación pensado para sustituir a HTML como estándar para las páginas Web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones más estrictas de XML.

XML: es el acrónimo del inglés eXtensible Markup Language (Lenguaje Extensible de Marcas) desarrollado por el World Wide Web Consortium.

XPATH: lenguaje (basado en XML) que permite seleccionar subconjuntos de un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto plano. XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML.

XSL: utilizado para transformar documentos XML en diferentes tipos de documentos como HTML, PDF, documentos de texto, entre otros.