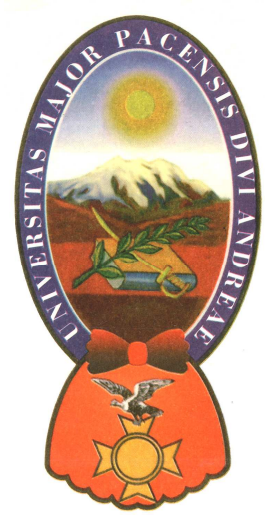
UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMATICA



APLICACIÓN DE JAVA 2 ENTERPRISE EDITION PARA SISTEMAS DISTRIBUIDOS

CASO SISTEMA DE GESTIONES Y SISTEMA ACADEMICO DE ODONTOLOGIA

PROYECTO DE GRADO

PARA OPTAR AL TITULO DE LICENCIATURA EN INFORMATICA

MENCION: INGENIERIA DE SISTEMAS INFORMATICOS

POSTULANTE: LUIS MARIO CONDORI QUISPE

TUTOR: Lic. Efraín Silva Sánchez

REVISOR: Lic. Carlos Mullisaca Choque

LA PAZ – BOLIVIA
2007

DEDICATORIA

A DIOS, mis PADRES, HERMANOS, y mi FAMILIA, por darme la oportunidad de estudiar y demostrar que uno puede salir adelante, siempre y cuando uno se lo propone y más aun con el apoyo de los seres queridos.

AGRADECIMIENTOS

Mis sinceros agradecimientos:

A nuestra Casa Superior de Estudios, Universidad Mayor de San Andrés, que en nuestra vida académica fue nuestro segundo hogar, además de formarme a nivel superior.

Al Lic. Efraín Silva Sánchez por su apoyo y confianza en mi persona para poder culminar este proyecto

Al Lic. Carlos Mullisaca Choque por sus consejos y recomendaciones para la pronta culminación de mi trabajo.

Al Lic. Ivan Vallejos por su tiempo y disposición, sin la cual no hubiera sido posible la realización de este proyecto.

A todos mis compañeros de trabajo y compañeros de la Universidad por el apoyo y por los momentos gratos en la tapa de estudios.

RESUMEN

El objetivo del Proyecto de Grado, es de poder compartir información entre dos sistemas que están físicamente distribuidos, utilizando los conceptos de Sistemas Distribuidos y la aplicación de un lenguaje de Programación Empresarial como es el J2EE.

Este proyecto surgió de la necesidad de compartir información entre dos sistemas como son: el Sistema de Gestiones de la Universidad Mayor de San Andrés con el Sistema de Seguimiento Académico de la Facultad de Odontología. La información que se transmite entre ambos sistemas se realiza con procesos manuales de migración de datos.

Este proyecto se resume en un modulo de Transacciones independiente de los sistemas, que permite el intercambio de información entre los dos sistemas, además que solo es enviada y recibida la información seleccionada de acuerdo a los requerimientos de ambos sistemas.

Entre los alcances del proyecto de grado esta poder aplicar este proyecto, es decir este modulo de Transacciones a las demás facultades para evitar los procesos manuales que se realizan de una Base de Datos a otra.

INDICE GENERAL

PÁGINA

CAPITULO I INTRODUCCION

CAPITULO II MARCO TEORICO

Anexos

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO I
INTRODUCCION

1.1 INTRODUCCION

Es evidente que cuanto más crece el conocimiento, avanza mas la tecnología, especialmente en la informática que hoy en día se incorpora al desarrollo de todas las instituciones sean estas publicas o privadas, esté propósito de incorporar la informática en las instituciones tiene como resultado final, brindar un buen servicio a los usuarios finales.

La **UMSA (Universidad Mayor de San Andrés),** institución Autónoma formadora de profesionales no puede estar al margen del avance de la tecnología, es por esta razón que tiene la imperiosa necesidad de adoptar mecanismos de automatización de sus procesos en sus distintas áreas.

En la actualidad el Programa UMSATIC a través del área de Sistemas de Información y con la colaboración del Centro de Procesamiento de Datos e Información (CPDI), a desarrollado dos sistemas de información:

- Sistema de Gestiones (Matriculación)
- Sistema de Información Académica

Estos dos sistemas están desarrollados en Java 2 Enterprise Edition (J2EE). El Sistema de Gestiones controla la matriculación de alumnos nuevos, matriculación de Postgrado, prorrogas, etc. El Sistema de Información Académico controla los procesos académicos de las distintas facultades y sus respectivas carreras de la UMSA.

El presente trabajo pretende realizar una Aplicación para Sistemas Distribuidos en este caso el Sistema de Gestiones y el Sistema de Información Académico de la Facultad de Odontología, esta facultad ya cuenta con el nuevo Sistema de Información Académico, por tanto esta facultad esta apto para poder realizar esta Aplicación para Sistemas Distribuidos y compartir información con el de Sistema de Gestiones de la UMSA.

Actualmente la actualización de la información entre Sistema de Gestiones con el sistema de Información Académico de la Facultad de Odontología se realiza mediante procesos manuales de migración de datos, por lo tanto se hace necesario realizar la aplicación de Base de Datos Distribuida que permita actualizar los datos en tiempo real.

En el Capitulo I plantemos la problemática, los objetivos, limites, alcances además de los métodos que se harán uso en el desarrollo de este trabajo. En el Capitulo II Tenemos el Marco Teórico donde describimos las herramientas que utilizamos para el logro de nuestro objetivos. El Capitulo III es el Desarrollo de del Proyecto de Grado utilizando las tecnologías presentadas en le capitulo anterior y el Capitulo IV presentamos nuestras Conclusiones y Recomendaciones.

1.2 ANTECEDENTES

Actualmente en nuestro medio no existen trabajos similares o proyectos parecidos al que se plantea. Pero si existe la aplicación de Base de Datos Distribuidas en otro tipo de aplicaciones, nuestros estudios nos llevan a los siguientes proyectos y/o tesis:

- El proyecto presentado por Edgar Navarro Guitierrez (2002): "Tratamiento de Apoyo al Registro Administrativo Sobre la Plataforma Internet y Base de Datos Distribuida, Caso Instituto Nacional de Estadística".

- La tesis presentada por Glasinovic y Landivar (1997): "Redes de Comunicación y Base de Datos Distribuidas Sistema de Bancos".

1.3 PLANTEAMIENTO DEL PROBLEMA

"La utilización de dos sistemas que tienen propósitos diferentes pero requieren información uno del otro y viceversa, además de estar distribuidos geográficamente. Lleva a realizar procesos manuales en la actualización de datos entre los dos Sistemas de Información, este proceso manual tiene como consecuencia el retraso en la información que es requerida por ambos sistemas."

Este proyecto de grado Aplicación de Java 2 Enterprice Edition para Sistemas Distribuidos Caso Sistema de Gestiones y Sistema Académico de Odontología, será capaz de realizar el intercambio de la información entre los dos sistemas en tiempo real con la ayuda de la red interna o el Internet con el que se cuenta actualmente.

Los principales problemas que serán encarados son:

- El acceso a la información generada del Sistema de Gestiones por parte de Sistema de Información Académico es de forma manual.

- El acceso a la información generada del Sistema de Información Académico de la Facultad de Odontología por parte del Sistema de Gestiones se la realiza pro procesos manuales.

- Incertidumbre de los alumnos matriculados e inscritos.

- Proceso de migración de un sistema a otro se realiza de forma manual.

- El enlace directo entre los dos Sistemas de Información es vía consultas, con migraciones de datos manuales.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

"Desarrollar un Servicio Web para Sistemas Distribuidos. Entre la Administración de la UMSA y el Sistema Académico de Facultad de Odontología".

1.4.2 OBJETIVOS ESPECIFICOS

- Desarrollar el Servicio Web en el Sistema de Gestiones para el envío y recepción de información.

- Desarrollar el Servicio Web en el Sistema Información Académico para el envío y recepción de información.

- Aplicar seguridad informática en los módulos de envío y recepción de datos para resguardar los datos que serán enviados y recibidos.

- Realizar una conexión directa entre los Sistemas de Gestiones y el Sistema de Información Académico de la Facultad de Odontología.

1.5 JUSTIFICACIÓN

El uso de alta tecnología de comunicación como es las redes de computadoras mejora los métodos de acceso, manipulación y seguridad de los datos. También es factible por que el proyecto puede desarrollarse con los equipos existentes en la institución.

Además, de implementar la comunicación entre el Sistema de Gestiones y el Sistema de Académico será de gran beneficio para la UMSA, por cuanto podrá contar con una nueva herramienta que agilicen sus procesos de actualización de información.

Tomando siempre en cuanta que la información es un recurso que debe ser tratado correctamente debemos comprender que existen costos asociados con la producción, distribución, seguridad, almacenamiento y recuperación de la información, aunque la información se encuentre a nuestro alrededor esta no es gratuita y su uso por lo tanto debe ser tratada de la mejor forma posible a favor de la UMSA.

La propuesta planteada contempla la utilización de los equipos ya existentes, lo que significa que no habrá gastos extras a la hora de la implementación de leste servicio Web.

Debido al crecimiento de la población estudiantil de la UMSA, como institución autónoma formadora de profesionales, es necesario mencionar que se desarrollan varios procesos vitales, estos procesos manejan información que deben ser conocidos en tiempo real para la toma de decisiones, tanto de los administrativos como de los estudiantes de la UMSA.

1.6 LIMITES Y ALCANCES

Este proyecto de grado Aplicación de Java 2 Enterprise Edition para Sistemas Distribuidos. Caso Sistema de Gestiones y Sistema Académico de Odontología, se limita a realizar la especificación de requerimientos de acuerdo a

la necesidad de los sistemas en el envío de información entre los Sistemas de Gestiones y el Sistema Académico de Odontología.

Se esta tomando en cuenta la Aplicación de Java 2 Enterprise Edition para Sistemas Distribuidos. Caso Sistema de Gestiones y el Sistema Académico de Odontología, como base fundamental para la implementación en los demás sistemas académicos de las distintas carreras con las que cuenta la UMSA.

1.7 METODO

La metodología escogida para el desarrollo del proyecto se basa en etapas y actividades de desarrollo Orientado a Objetos basado en UML.

El Lenguaje Unificado de Modelado (UML) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

Las etapas con las que cuenta esta metodología son:

- Levantamiento de Requerimientos
- Análisis de Requerimientos
- Diseño Detallado
- Diseño e Implantación de la Base de Datos
- Implantación y Pruebas

CAPITULO II
MARCO TEORICO

En este capitulo se describen los puntos necesarios para comprender el funcionamiento de los Sistemas Distribuidos y la aplicación de el Java 2 Enterprise Edition (J2EE) en estos Sistemas Distribuidos. Los temas que se tratarán son: definición de Sistemas Distribuidos, Web Services principales componentes que pueden conformarlo, utilidades que ofrece, Axis, javamail, etc. Definición de Base de Datos Distribuidas principales componentes. Modelo de Programación Orientado a Objetos. El Lenguaje de programación J2EE. El Sistema Gestor de Base de Datos usado en el desarrollo del proyecto (Postgres).

Buscaremos esclarecer los conceptos más importantes relacionados con los Web Services y Sistemas Distribuidos y la aplicación de un lenguaje de programación como es el J2EE en estos sistemas, además se debe comprender el desarrollo realizado en el proyecto que se explicara durante todo el todo el trabajo.

2.1 SISTEMAS DISTRIBUIDOS

Son sistemas cuyos componentes hardware y software, están en ordenadores conectados en red, se comunican y coordinan sus acciones mediante el paso de mensajes, para el logro de un objetivo, se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidor.[2]

2.1.1 CARACTERISTICAS DE LOS SISTEMAS DISTRIBUIDOS:

- Concurrencia.- **Esta característica de los Sistemas Distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan con la red.[2]**

- Carencia de reloj global.- **Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tiene una temporalización general, esta mas bien distribuida a los componentes.[2]**

- Fallos independientes de los componentes.- **Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando.[2]**

Una parte fundamental de los sistemas distribuidos es la computación Cliente – Servidor. Este modelo es el que predomina en la actualidad, permite descentralizar el procesamiento y recursos, sobre todo, de cada uno de los servicios y de la visualización de la interfaz grafica del usuario. Esto hace que ciertos servidores estén dedicados solo a una aplicación determinada y por tanto cada aplicación se ejecuta de una forma eficiente.

A continuación se presenta una lista de los servidores más comunes:

2.1.2 CATEGORIAS DE SERVIDORES

- Servidores de archivos.- **proporciona archivos para clientes. Si los archivos no fueran tan grandes y los usuarios que comparten esos archivos no fueran muchos, esto seria una gran opción de almacenamiento y procesamiento de archivos. El cliente solicita los archivos y el servidor los ubica y los envía.**

  Ejemplo, una aplicación que requiera el almacenamiento y acceso a dibujos técnicos, digamos para una empresa de fabricación utilizaría un servidor de archivos para almacenar y proporcionar los dibujos para los clientes.[3]

- Servidores de Base de Datos.- **Son los que almacenan gran cantidad de datos estructurados, se diferencian de los de archivos pues la información que se envía ya es resumida en la Base de Datos.**

  Ejemplo: El cliente hace una consulta, el servidor recibe esa consulta y extrae solo la información pertinente y envía esa respuesta al cliente.[1]

- Servidores de Software de Grupo.- **El software de grupo es aquel que permite organizar el trabajo de un grupo. El servidor gestiona los datos que dan soporte a estas tareas.**

  Ejemplo almacenar las listas de correo electrónico. El cliente puede indicarle, que se ha terminado una tarea y el servidor se lo envía al resto del grupo. [1]

- Servidores WEB.- **Los documentos Web se almacenan como paginas en una computadora conocida como servidor Web (browser) para ver las paginas Web normalmente pincha sobre un enlace en un documento Web existente. Esto dará como resultado un mensaje que se enviará al servidor Web que contiene la pagina. Este servidor responderá entonces enviando una página a su computadora, donde el navegador pueda visualizarlo.**

  De esta manera los servidores Web actúan como una forma de servidor de archivos, administrando archivos relativamente pequeños a usuarios quienes entonces utilizan un navegador para examinar estas paginas.[3]

- Servidores de Aplicación.- **Un servidor de aplicación se dedica a una aplicación única. Tales servidores suelen escribirse utilizando un lenguaje tal como Java o C++.**

  Ejemplo típico del servidor que se utiliza en el dibujo de un fabricante de aviones que gestionaba las versiones diferentes de dibujos producidos por el personal técnico iría dirigido a algún proceso de fabricación.[3] entonces básicamente es una aplicación a la que pueden acceder los clientes.

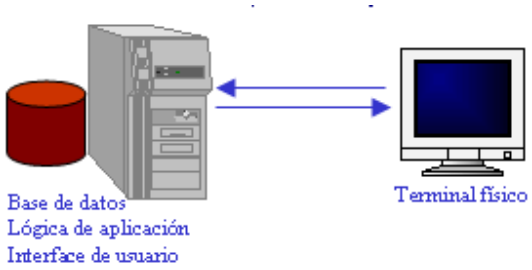
2.1.3 CLASIFICACIÓN DE LOS SISTEMAS CLIENTE SERVIDOR

A continuación mostramos la clasificación de los sistemas cliente/servidor de acuerdo al nivel de abstracción del servicio que ofrecen:

a) REPRESENTACIÓN DISTRIBUIDA

La interacción con el usuario se realiza en el servidor, el cliente hace de pasarela entre el usuario y el servidor. [4]

Figura 2.1: Representación Distribuida

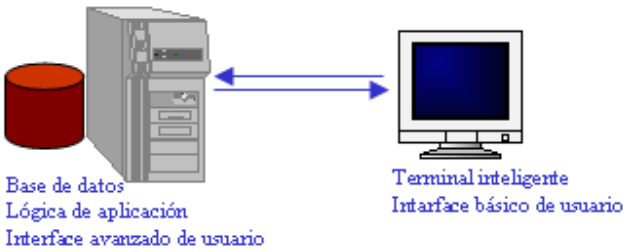


Fuente: Sistemas Distribuidos 2006. Univ. Zaragoza

b) REPRESENTACIÓN REMOTA

La lógica de la aplicación y la base de datos se encuentran en el servidor, el cliente recibe y formatea los datos para interactuar con el usuario. [4]

Figura 2.2: Representación Remota

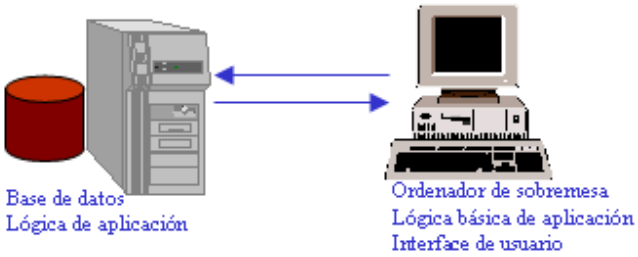


Fuente: Sistemas Distribuidos 2006. Univ. Zaragoza

c) LÓGICA DISTRIBUIDA

El cliente se encarga de la interacción con el usuario y de algunas funciones triviales de la aplicación. Por ejemplo controles de rangos de campos, campos obligatorios, etc. Mientras el resto de la aplicación, junto con la base de datos están en el servidor. [4]

Figura 2.3: Lógica Distribuida

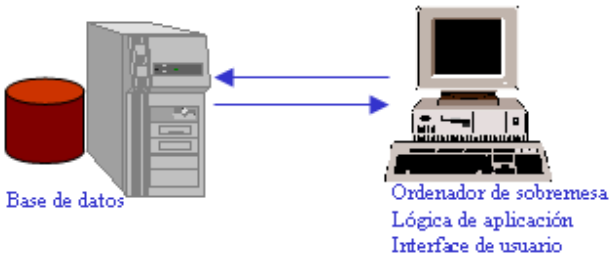


Fuente: Sistemas Distribuidos 2006. Univ. Zaragoza

d) GESTIÓN REMOTA DE DATOS

El cliente realiza la interacción con el usuario y ejecuta la aplicación y el servidor es quien maneja los datos. [4]

Figura 2.4: Gestión Remota de Datos

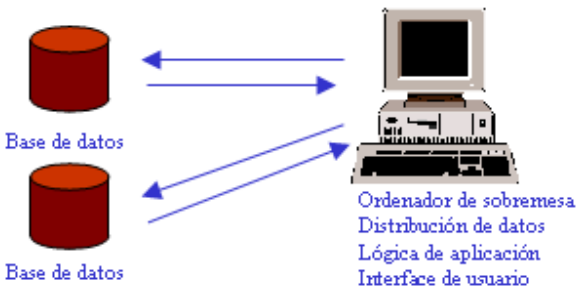


Fuente: Sistemas Distribuidos 2006. Univ. Zaragoza

e) BASE DE DATOS DISTRIBUIDAS

El cliente realiza la interacción con el usuario, ejecuta la aplicación debe conocer la tecnología de la red, así como la disposición y ubicación de los datos se delega parte de la gestión de la base de datos al cliente. [4]

Figura 2.5: Base de Datos Distribuida

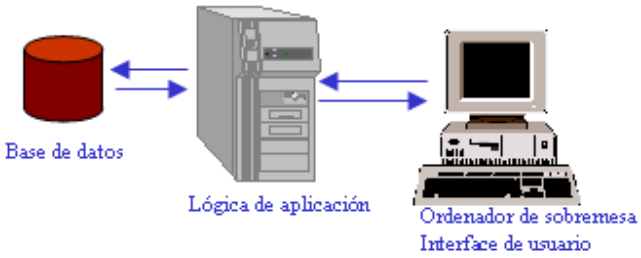


Fuente: Sistemas Distribuidos 2006. Univ. Zaragoza

f) CLIENTE SERVIDOR A TRES NIVELES

El cliente se encarga de la interacción con el usuario, el servidor de la lógica de aplicación y la base de datos puede estar en otro servidor. [4]

Figura 2.6: Cliente Servidor a Tres Niveles



Fuente: Sistemas Distribuidos 2006. Univ. Zaragoza

En las aplicaciones distribuidas una parte fundamental son las Bases de Datos de cada aplicación estas Bases de Datos llegan a ser distribuidas, las Bases de Datos Distribuidas tienen características que describimos a continuación:

2.2 BASE DE DATOS DISTRIBUIDA

Es una colección de datos (base de datos) construida sobre una red y que pertenecen, lógicamente, a un solo sistema distribuido, la cual cumple las siguientes condiciones:

- La información de la base de datos esta almacenada físicamente en diferentes sitios de la red.

- En cada sitio de la red, la parte de la información, se constituye como una base de datos en sí misma.

- Las bases de datos locales tienen sus propios usuarios locales, sus propios DBMS y programas para Administración de transacciones, además de su propio administrador local de comunicación de datos.

- Estas base de datos locales deben de tener una extensión, que gestione las funciones de sociedad necesarias; la combinación de estos componentes con los sistemas de administración de base de datos locales, es lo que se conoce como Sistema Administrador de Base de Datos Distribuidas.

- Este gestor global permite que usuarios puedan acceder a los datos desde cualquier punto de la red, como si lo hicieran con los datos de su base de datos local, es decir, para el usuario, no debe existir diferencia en trabajar con datos locales o datos de otros sitios de la red.

En consecuencia, la base de datos distribuida, es como una unidad virtual, cuyas partes se almacenan físicamente en varias bases de datos “reales" distintas, ubicadas en diferentes sitios.

En la situación actual de la institución, el uso de los sistemas distribuidos que son actualizados a nivel de Bases de Datos y con procesos manuales de consulta utilizando el SQL. Entonces para evitar estos procesos manuales se propone realizar servicios web o Web Services.

2.3 DESARROLLO WEB

Caso particular de los sistemas Cliente-Servidor con representación remota. En donde se dispone de un protocolo estándar: HTTP y un Middleware denominado WebServer. En la actualidad la aplicación de sistemas informáticos basados en Internet, es una herramienta fundamental para las organizaciones que desean tener cierta presencia competitiva. [6]

Tecnologías de la lógica de la aplicación en el servidor Web

- **CGI**: Common Gateware Interface..- Son programas que se ejecutan en el servidor, pueden servir como pasarela con una aplicación o base de datos o para generar documentos HTML de forma automática. Cada petición http ejecuta un proceso, el cual analiza la solicitud y genera un resultado. Son independientes del SO, y presentan la ventaja de que, dado un programa escrito en un lenguaje cualquiera, es fácil adaptarlo a un CGI. Entre los lenguajes que se usan para CGI´s, el más popular es el **Perl**.

- **Servlets**: Pequeños programas en Java que se ejecutan de forma persistente en el servidor, y que, por lo tanto, tienen una activación muy rápida, y una forma más simple de hacerlo. Estos programas procesan una petición y generan la página de respuesta.

- **JSP** (*Java Server Pages*), que consisten en pequeños trozos de código en Java que se insertan dentro de páginas web, de forma análoga a los ASPs. Ambas opciones, hoy en día, son muy populares en sitios de comercio.

Frente a los ASPs, la ventaja que presentan es que son independientes del sistema operativo y del procesador de la máquina.

2.4 SERVICIOS WEB (Web Services)

Un servicio Web (o en ingles Web Services) es una colección de protocolos y estándares que sirven para intercambiar datos entre las aplicaciones. Distintas aplicaciones de software desarrolladas en distintos lenguajes de programación, diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

Un Web Services es un contenedor que encapsula funciones específicas y hace que estas funciones puedan ser utilizadas en otros servidores, alguna ventajas que presentan son:

- **Son programables.**
- **Están basados en XML, que es un lenguaje abierto.**
- **Son autos descriptivos.**
- **Pueden buscar registros de otros Web Services.**

2.4.1 TECNOLOGIAS QUE SE USAN EN LOS SERVICIOS WEB

- **Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de servicio Web.**

- **XML (Lenguaje Extensible de Etiquetas). Es el formato estándar para describir datos y crear etiquetas que se vayan a intercambiar. Las características especiales son la independencia de datos, o de la separación de los contenidos de su presentación. Es un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos.**

  Los documentos XML se componen de unidades de almacenamiento llamadas entidades, que contienen datos analizados o sin analizar, los datos analizados se componen de caracteres, algunos de los cuales forman los datos del documento y el resto forman las etiquetas.
  Las etiquetas codifican la descripción de la estructura lógica y de almacenamiento del documento, XML proporciona un mecanismo para imponer restricciones en la estructura lógica y de almacenamiento.

- **SOAP (Protocolo de Acceso Simple a Objetos). o XML-RPC Protocolo sobre los que se establece el intercambio. Los mensajes deberían tener un formato determinado empleando XML para encapsular los parámetros de la petición, el mensaje esta compuesto de tres partes: un sobre, encabezado y el cuerpo.**

  El sobre envuelve al mensaje y contiene al encabezado y el cuerpo; el encabezado es un elemento opcional que provee información para el enrutamiento del mensaje; el cuerpo contiene datos etiquetados como el XML.

- **Otros protocolos: los datos en XML también pueden variarse de una aplicación a otra mediante protocolos normales como HTTP, FTP, o SMTP.**

- **WSDL (Lenguaje de descripción de servicios Web). Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos necesarios para establecer una comunicación con los servicios Web.**

- **UDDI (Descripción, Descubrimiento e integración Universal). Protocolo para publicar la información de los servicios Web. Permite a las aplicaciones comprobar que servicios web están disponibles. Provee un mecanismo para que los negocios se describan a si mismo y los tipos de servicios que proporcionan y luego se pueden registrar y publicarse en un registro UDDI.**

  Tales negocios pueden ser buscados, consultados o descubiertos por otros negocios mensajes SOAP.

- **WS – Security (Web Services Security). Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.**

2.4.2 VENTAJAS DE LOS SERVICIOS WEB

- **Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o plataformas sobre las que se instalen.**

- **Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen mas fácil de acceder su contenido y atender su funcionamiento.**

- **Al apoyarse en HTTP, los servicios web pueden aprovecharse de los sistemas de seguridad firwall sin necesidad de cambiar las reglas de filtrado.**

- **Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos pueden ser combinados fácilmente para proveer servicios integrados.**

- **Permiten la interoperabilidad entre las plataformas de distintos fabricantes por medio de protocolos estándar.**

2.4.3 RAZONES PARA CREAR SERVICIOS WEB

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls que filtran y protegen gran parte del tráfico de Internet. Cierran casi todos los puertos TCP salvo el 80, que es, precisamente el que usan los navegadores. Los servicios Web se vehiculan por este puerto, por la simple razón de que no resultan bloqueados.

Otra razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante,

dado que la independencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más grande.

Uno de los servidores de aplicaciones para servidores Web es el Axis se le considera como plataforma para crear nuestro Web Services.

2.4.4 AXIS

AXIS es una herramienta open source en Java que permite la construcción y desarrollo de servicios Web así como de clientes que puedan acceder a ellos. Gracias a esta herramienta se simplifica la tarea de dejar visibles una serie de métodos de objetos JAVA, pasando estos métodos a ser un servicio disponible que pueden utilizar los diferentes clientes.

Axis proporciona:

- **Un entorno de ejecución para Servicios Web Java (*.jws).**
- **Herramientas para crear WSDL desde clases java.**
- **Herramientas para crear clientes Java desde un WSDL.**
- **Herramientas para desplegar, probar y monitorizar Servicios Web.**
- **Integración con servicios de aplicaciones y contenedores de Servelets.**

Además el Axis tiene las siguientes características:

- **Velocidad.- El Axis utiliza el SAX, que es un analizador que cuenta con la velocidad significativa para versiones del SOAP de Apache.**

- **Flexibilidad.- La arquitectura del Axis otorga libertad completa a la hora de insertar extensiones cuando las tareas se procesa o se desarrolla los sistemas.**

- **Estabilidad.- Axis define un sistema de interfaces públicas que cambian relativamente y comparan con el resto del Axis.**

- **Despliegue Componente – Orientado.- Puede definir fácilmente las redes de trabajo que sean reutilizables o implementar políticas para el proceso común de las aplicaciones o distribuir a clientes.**

- **Transporte de Framework.- Tiene una limpia y simple abstracción para diseñar transportes (es decir los que envían y los que reciben para SOAP o los otros protocolos tales como SMTP, FTP, mensajes – orientados midleware, etc) y la base del proceso es completamente independientemente al transporte.**

- **Soporta WSDL.- Axis soporta WSDL versión 1.1 permite construir accesos remotos a servicios, también permite exportar automáticamente descripciones legibles para servicios en Axis.**

  El requerimiento necesario para su funcionamiento de Axis para Java es el Tomcat.

2.4.5 JAVAMAIL

El JavaMail es una expansión de Java que facilita el envió y recepción de e-mail desde código Java. Su principal característica es que JavaMail implementa el protocolo SMTP así como los distintos tipos de conexión con servidores de correos, autentificación con usuario y password.

2.4.6 TOMCAT

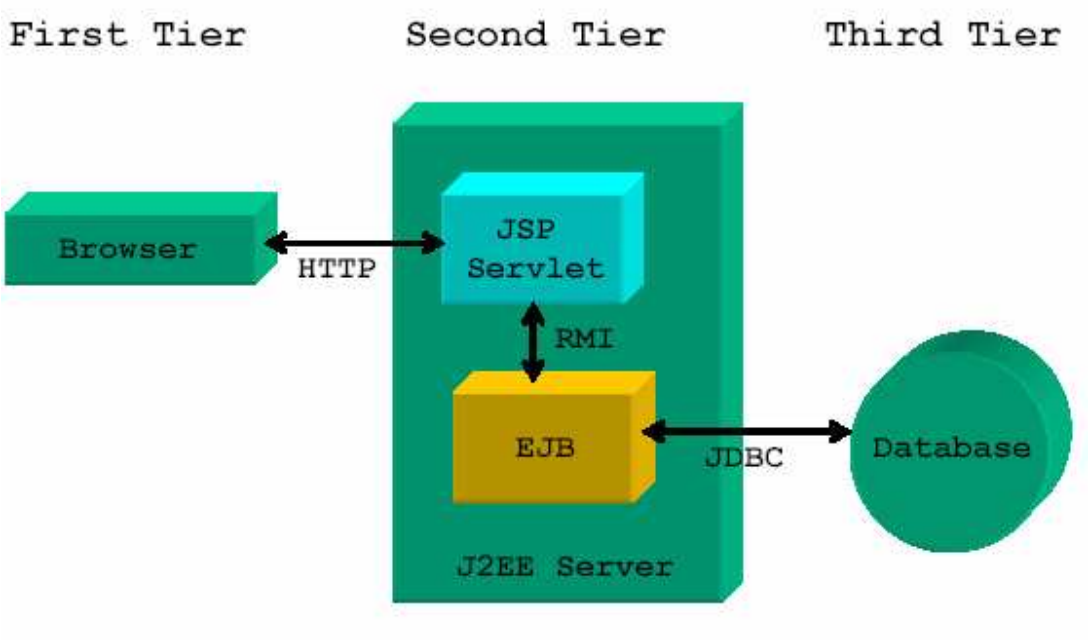
Tomcat es un Servidor Web de Apache. Para poder realizar Servicios Web necesitamos que estos reposen sobre un servidor Web. Tomcat en particular es uno de ellos. Tomcat es una implementación libre de código abierto, fue desarrollado en el proyecto Jakarta en Apache Software Fundación. También se puede decir que es un contenedor de servlet y JavaServer Pages.

2.4.7 J2EE: JAVA 2 EDICION EMPRESARIAL (Java 2 Eterprise Edition)

El J2EE es la evolución de Java, es una tecnología exclusivamente empresarial, orientada hacia un tipo de desarrollo muy concreto, J2EE esta especializado en aplicaciones WEB Todas sus librerías, objetos e interfaces están orientados a este tipo de aplicaciones. J2EE ofrece un estándar a la hora de desarrollar componentes de reglas de negocio y componentes de acceso a datos.[7]

La arquitectura del J2EE es como se muestra en la siguiente figura:

Figura 2.8: Arquitectura del J2EE



Fuente: Desarrollo Ágil con J2EE 2006 OpenSource

2.5 OBJETOS DISTRIBUIDOS

En los sistemas Cliente/Servidor, un objeto distribuido es aquel que esta gestionando por un servidor y sus clientes invocan sus métodos utilizando un método de invocación remota. El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y

el resultado se devuelve al cliente en otro mensaje.[2] y las tres tecnologías importantes y mas usadas en este ámbito son:

2.5.1 RMI: INVOCACION REMOTA DE METODOS (Remote Invocation Method)

Fue el primer fremeworks para crear sistemas distribuidos de Java. El sistema de Invocación Remota de Métodos (RMI) de Java permite, a un objeto que se esta ejecutando en una Máquina Virtual Java (JVM), llamar a métodos de otro objeto que esta en otra VM diferente. Esta tecnología esta asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.[2]

2.5.2 DCOM: MODELO DE OBJETO COMPONENTE DISTRIBUIDO (Distributed Component Object Model)

El Modelo de Objeto Componente Distribuido, esta incluido en los sistemas operativos de Microsoft. Es un juego de conceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios de objetos de programa de servidores en otros ordenadores dentro de una red. Esta tecnología esta asociada a la plataforma de productos de Microsoft. [2]

2.7 POSTGRESQL[8]

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relaciónales (ORDBMS) que ha sido desarrollado de varias formas desde 1977. Comenzó como un proyecto denominado Ingres en la Universidad Berkeley de California. Ingres fue más tarde desarrollado comercialmente por la Relational Technologies/Ingres Corporation.

En 1986 otro equipo dirigido por Michael Stonebraker de Berkeley continuó el desarrollo del código de Ingres para crear un sistema de bases de datos objeto-relacionales llamado Postgres.

En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, Postgres fue renombrado a PostgreSQL, tras un breve periplo como Postgres95. El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

> PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python).

Posee muchas características (las cuales serán discutidas en detalle en la sección denominada ”Juego de Características de PostgreSQL") que tradicionalmente sólo se podían ver en productos comerciales de alto calibre.[6]

2.7 Proceso Unificado de Desarrollo (RUP)

Para el proceso de desarrollo del software se utiliza el RUP para pequeños proyectos y se esta tomando en cuenta los siguientes puntos de este:

1. Fase de Inicio

    - Establecimiento de los requisitos para el WS.
    - Selección de la herramienta de software.

2. Fase de Elaboración

    - Creación del modelo de análisis del WS.
    - Integración del marco de trabajo (herramientas) para el desarrollo.
    - Creación del primer prototipo.
    - Creación del Modelo de Diseño del WS.
    - Creación de Modelos de Pruebas de Unidad y de Integración.
    - Creación del Modelo de Diseño de la aplicación cliente.

3. Fase de Construcción

- Creación del Modelo de Implementación.
- Implementación del WS y la aplicación cliente.
- Aplicación de pruebas de unidad e integración.

4. Fase de Transición
    - Ubicación del WS en el servidor final.
    - Promoción entre los usuarios potenciales.

CAPITULO III
DESARROLLO DE SISTEMAS DISTRIBUIDOS PARA EL SISTEMA DE GESTIONES Y SISTEMA ACADEMICO DE ODONTOLOGÍA

La uso de Sistemas Distribuidos en una institución grande como es la UMSA es beneficioso para esta institución, debido a que cuenta con muchas carreras donde la información es manejada de diferente manera en cada carrera, pero se debe considerar que la información entre los sistemas se la debe realizar en línea, es decir en tiempo real, los usuarios (estudiantes de la Universidad Mayor de San Andrés), son los directos involucrados en este sistema.

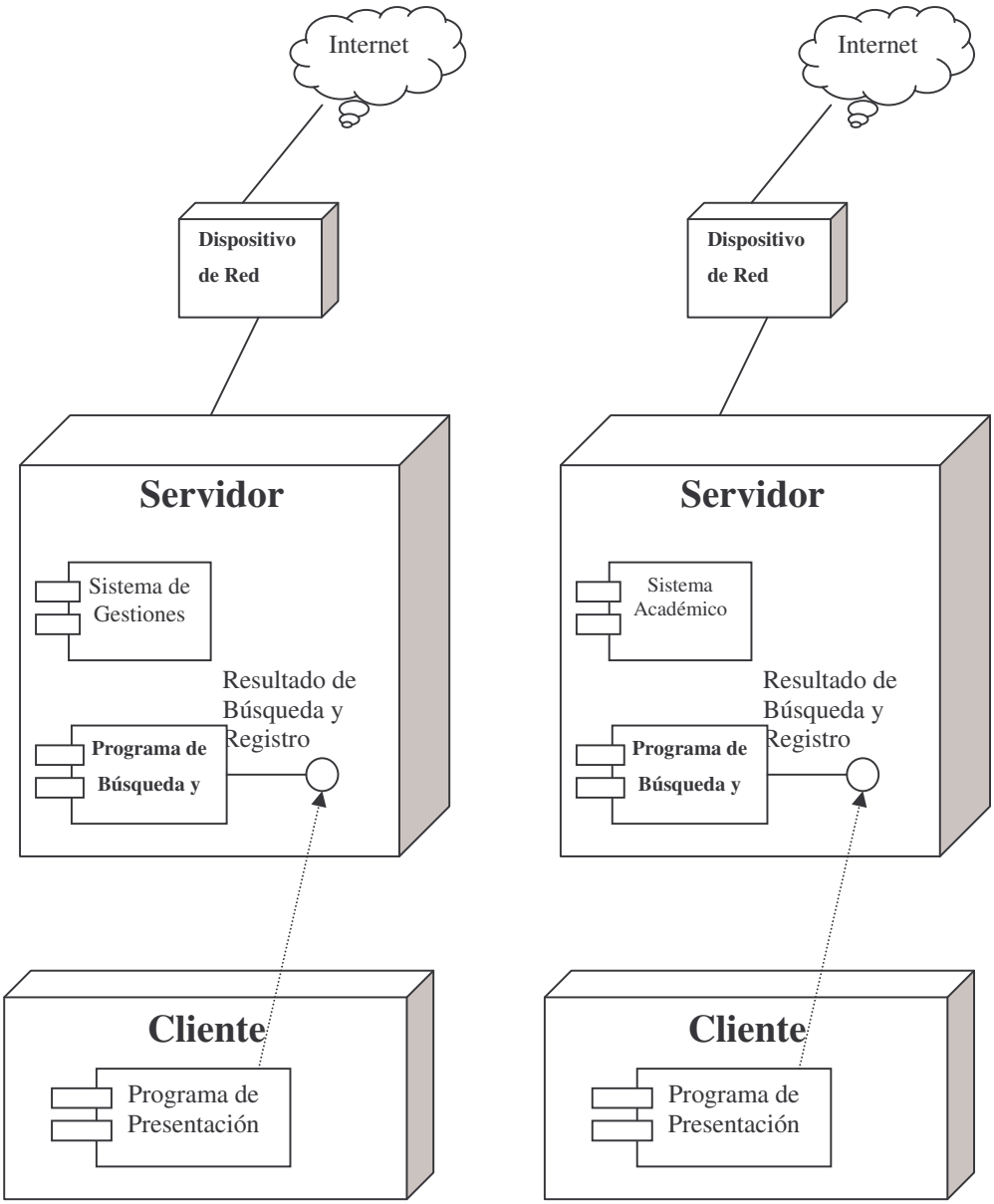
Debido a que requieren recabar información tanto desde su respectiva carrera como de Gestiones que es donde se centra la información. Es necesario mencionar que en estos momentos la información que se maneja es en forma separada, es decir que la información de gestiones tiene su propia Base de Datos y ocurre del mismo modo de las respectivas carreras de la UMSA.

Al realizar la petición de información desde gestiones a la carrera como la carrera de gestiones, este proceso se la realiza en forma manual atreves de consultas y migración de datos de una Base de Datos a la otra.

3.1 SITUACIÓN ACTUAL DE LOS SISTEMAS

Realizando el estudio respectivo de la documentación actual de los sistemas que involucran el proyecto, y verificar el estado de los actuales sistemas se pudo evidenciar que son dos sistemas que están distribuidos físicamente, y lo podemos representar como se muestra en la figura.

Figura 3.1: Modelo de la situación actual de los sistemas



Fuente: Elaboración propia

La principal necesidad es la unión de ambos sistemas, como se puede observar en el modelado (en la anterior figura), es el estado actual de los dos

sistemas, no tiene una vía de comunicación entre estos, y ambos requieren información uno del otro.

Actualmente el modo de cruce de información entre los dos sistemas se realiza con procesos manuales. Por tanto como requerimiento podemos decir que es la unión de estos dos sistemas aplicando Sistemas Distribuidos, para resolver este problema que se nos presenta.

3.2 CASOS DE USO DEL SISTEMA

Realizando un estudio minucioso de nuestro campo de trabajo y también entrevistando a las personas involucradas en estos sistemas pudimos identificar nuestros casos de uso.

Entonces nuestros casos de uso son:

- Inicio de sesión del administrador.
- Acceder a la información de Gestiones.
- Consultar información de Gestiones.
- Registrar información a Gestiones.
- Solicitud de Información de Gestiones.
- Armar solicitud.
- Transformar la solicitud.

- Registrar la Solicitud.
- Enviar la Datos transformados.
- Registrar datos.
- Enviar respuesta.
- Almacenar respuesta.
- Establecer el estado del envío.

En nuestros casos de uso solo tenemos un actor que es el **Administrador** del sistema tanto de Gestiones como el Administrador del sistema Académico de la facultad de Odontología, debemos entender que como Administrador están las personas que son encargados de operar el respectivo sistema.

Podemos apreciar dos tipos de casos de uso que son:

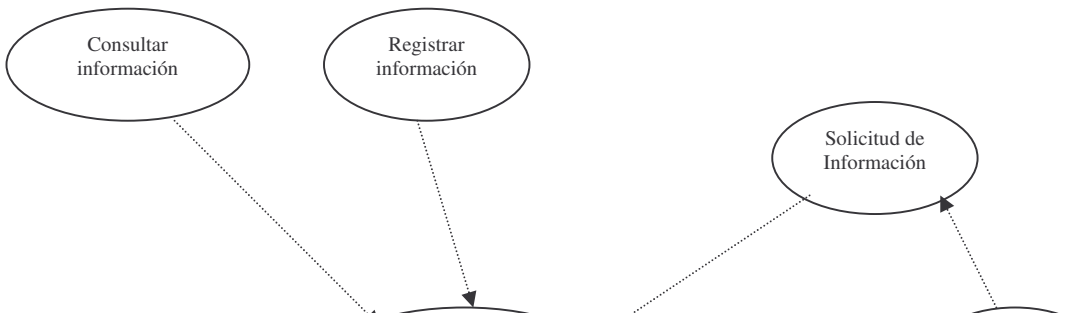
- Caso de uso del administrador del sistema de Gestiones.
- Caso de uso del Administrador del sistema Académico.

Los casos de uso tienen directa relación con los Administradores que son los directos encargados de registrar la información a la base de datos propia y además en la base de datos del

otro sistema, y viceversa, este envío de información a los otros sistemas se realiza mediante un evento donde el administrador podrá ejecutar este evento.

A continuación gratificamos los casos de uso

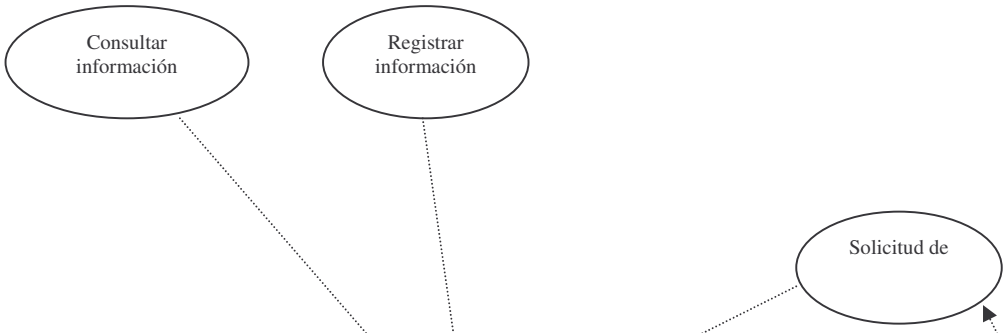
3.3 DIAGRAMAS DE CASO DE USO



Administrador

Gestiones

Figura 3.2: Diagrama de Caso de Uso del Administrador de Gestiones



Administrador

Académico

Figura 3.3: Diagrama de Caso de Uso del Administrador del Sistema Académico

3.4 DESCRIPCÍON DE LOS CASOS DE USO

Caso de uso: < 1 > Consultar Información

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Se consulta información de una cuenta de usuario.
Condición Final de Fallo: No se consulta la información de una cuenta de usuario.
Actor primario: Administrador de Gestiones.
Disparador: Consulta de la información de una cuenta de usuario.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: El administrador selecciona la opción de consultar la información del sistema para el usuario.
<PASO 2>: El sistema solicita por el campo por el cual se realizará la búsqueda.
<PASO 3>: El administrador introduce el valor del campo y solicita la búsqueda.
<PASO 4>: El sistema muestra el resultado de la búsqueda. Se despliega toda la información de la cuenta de usuario.

EXTENSIONES
<PASO 4a>: El sistema no muestra datos por no existir su datos del usuario en el sistema.

Caso de uso: < 2 > Registrar Información

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Se graba la información que el usuario registra.

Condición Final de Fallo: No registra la información que el de usuario quiere.

Actor primario: Administrador de Gestiones.

Disparador: El administrador registra de la información en la Base de Datos.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: El administrador selecciona la opción correspondiente para registrar la información de una cuenta de usuario.

<PASO 2>: El administrador introduce los datos en los campos necesarios para el registro de los nuevos datos.

<PASO 3>: El sistema solicita información para ver que los datos no sean duplicados y realizará la búsqueda por el campo que sea necesario.

<PASO 4>: El sistema muestra el resultado de la búsqueda y despliega toda la información del usuario, en caso de que el registro sea nuevo el sistema se pone listo para registrar los nuevos datos.

<PASO 5>: El administrador modifica todos los datos que se requiere o el administrador introduce los nuevos datos para el registro en la base de datos.

<PASO 6>: El sistema valida todos los datos que sean editados o registrados por primera ves del usuario.

<PASO 7>: Según a la validación de los datos el sistema muestra el botón de grabado de los nuevos datos en la Base de Datos.

EXTENSIONES

<PASO 6a>: En caso de que los datos que el administrador registra sean errados el sistema pedirá que sean corregidos estos datos.

<PASO 7a>: El sistema muestra un error al realizar el grabado de los datos por motivos extras al sistema.

3.5 DIAGRAMA DE SECUENCIAS

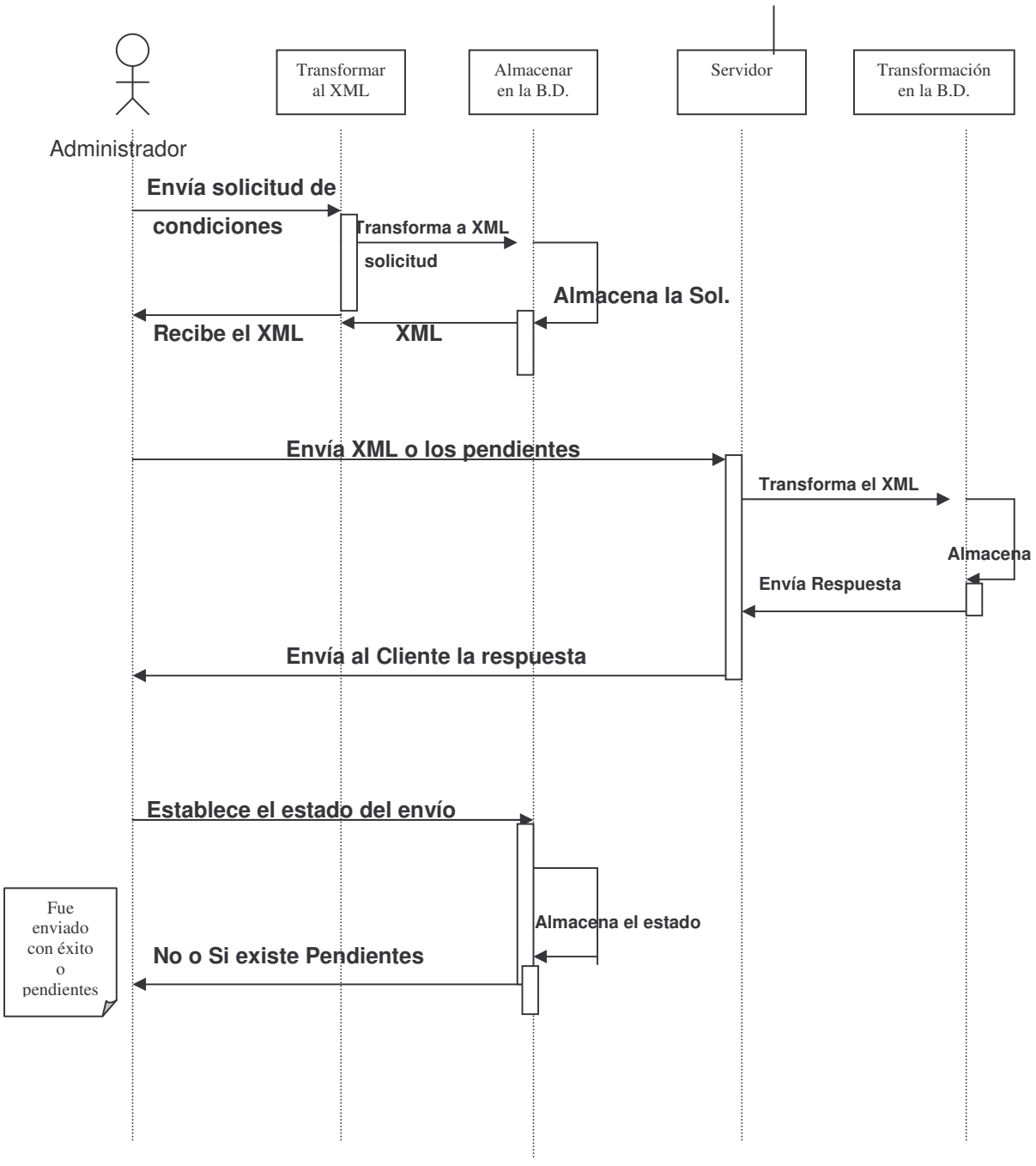
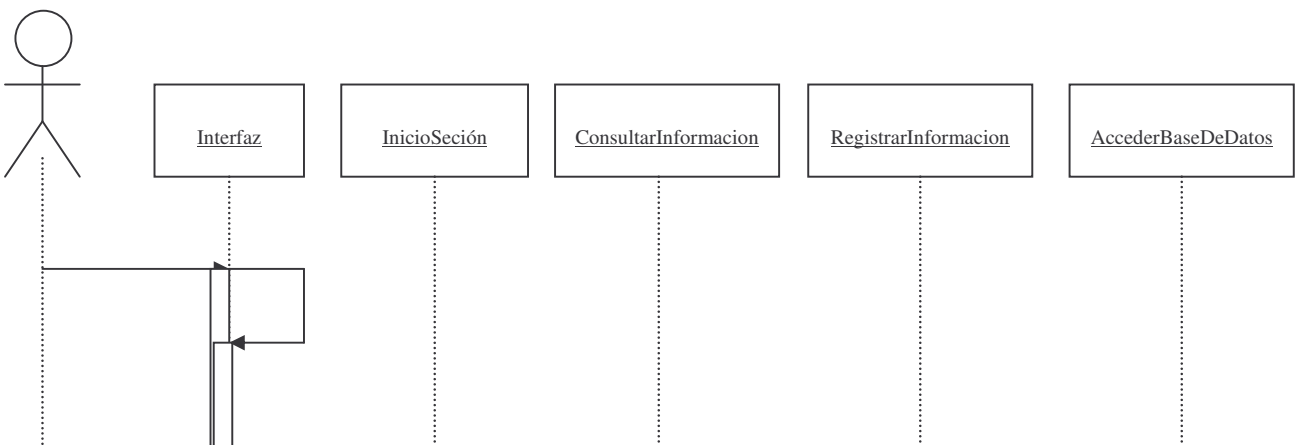


Figura 3.4: Diagrama de Secuencia General



Administradore

Ingreso

Validar Datos

Ingreso al Sistema

Retorno

Consulta Información

Activa Consulta

Verdadero

Retorna Información

Registra Información

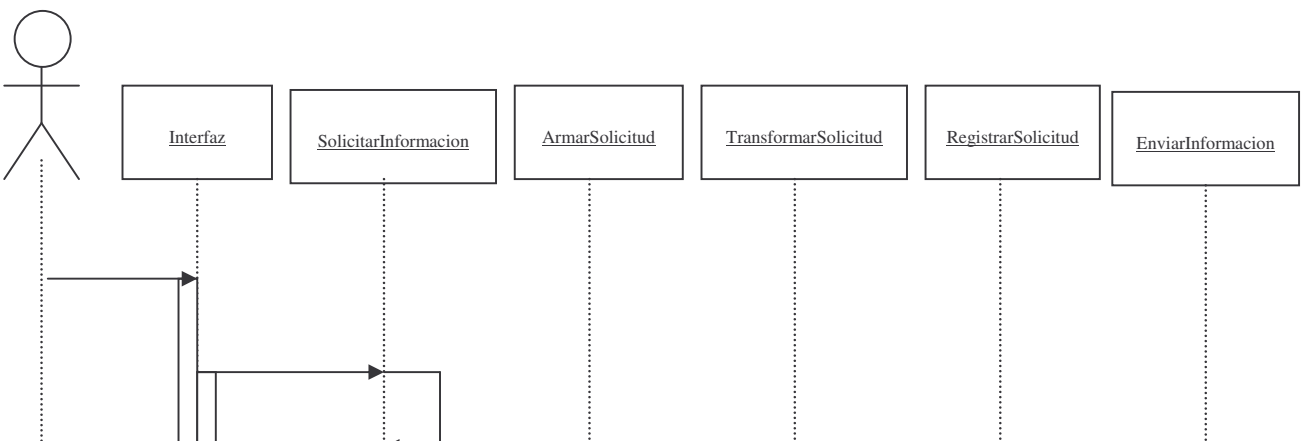
Activa Consulta

Verdadero

Retorna Información

Exito

Figura 3.5: Diagrama de Secuencias del Acceso, Consulta y Registro de la Información



Evento enviar

ActivarSolicitud

Buscar tipo de Solicitud

ActivarArmado

Estructura para
El Armado

ActivarTransformado

ActivarRegistro

Activar Query
De Registro

Enviar Inf.

Verdad

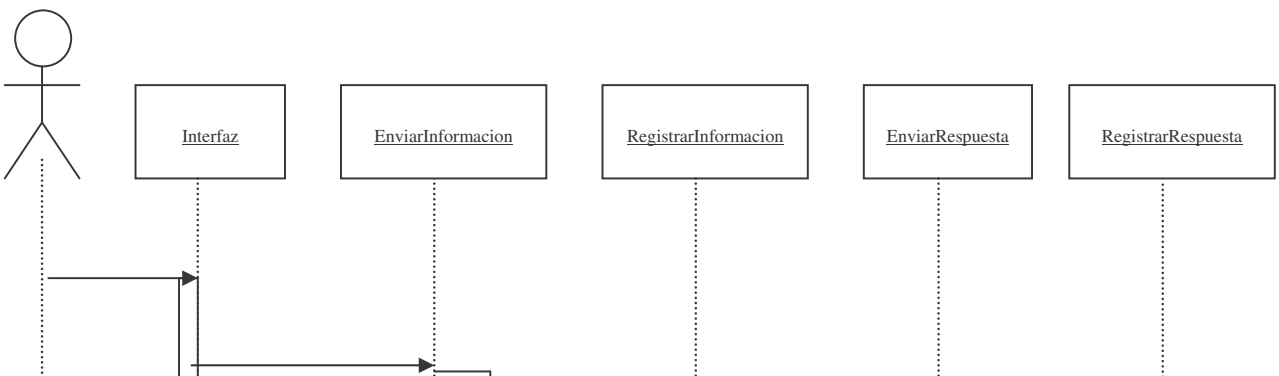
Cambiar estado

De Envío

Confirmación

Reporte

Figura 3.6: Diagrama de Secuencia del Envío de Información



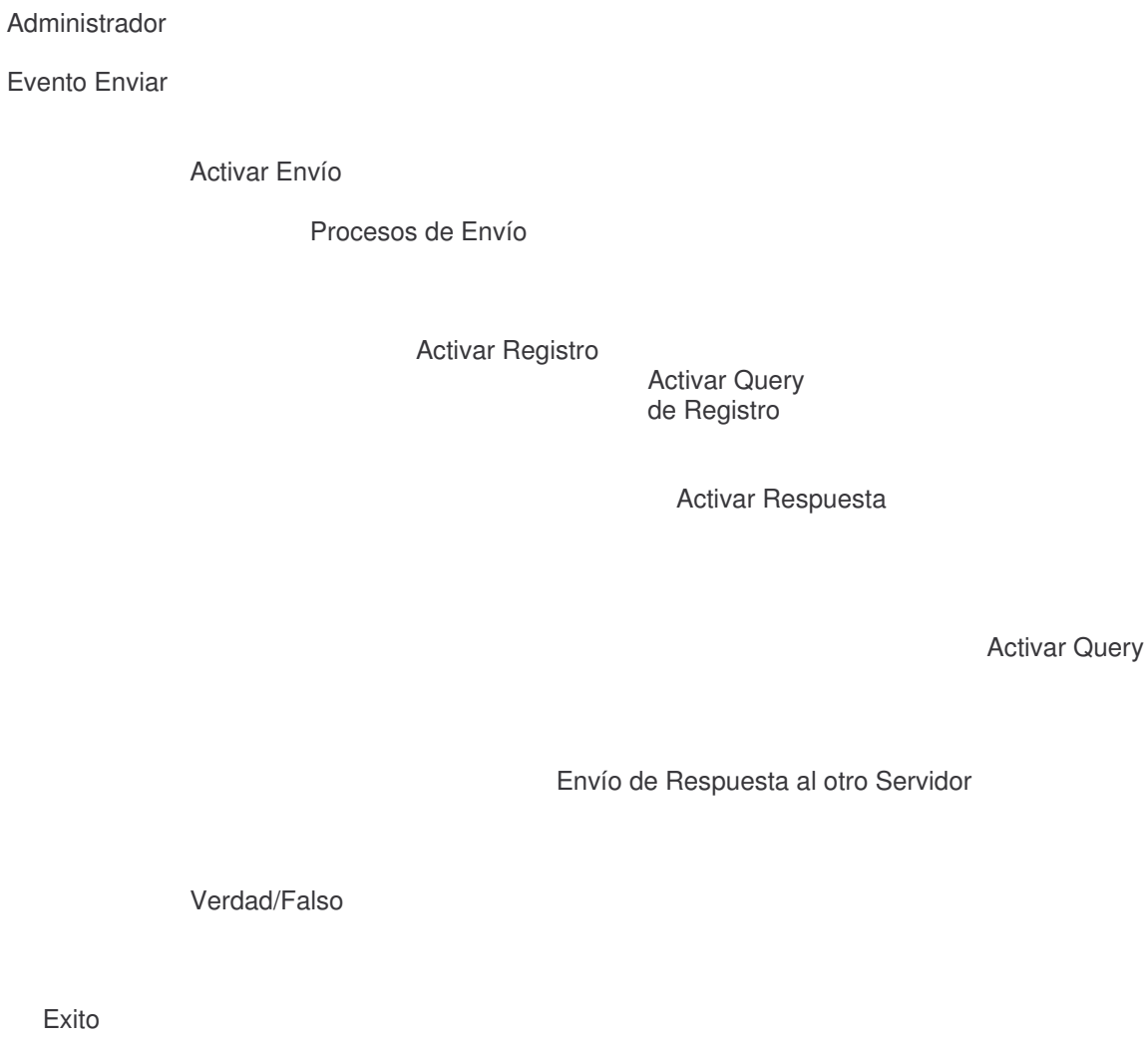


Figura 3.7: Diagrama de Secuencias del registro y Respuesta del Servidor

3.6 DIAGRAMA DE CLASES

Es necesario mencionar que normalmente los administradores de los sistemas o usuarios del sistema (operadores), los sistemas se desenvuelven en base a sus acciones de estos. Pero en este caso por el propósito del proyecto el administrador del sistema los usuarios del sistema pasan a segundo plano y damos énfasis en los procesos internos del los sistemas. Dada esta aclaración creamos nuestras clases:

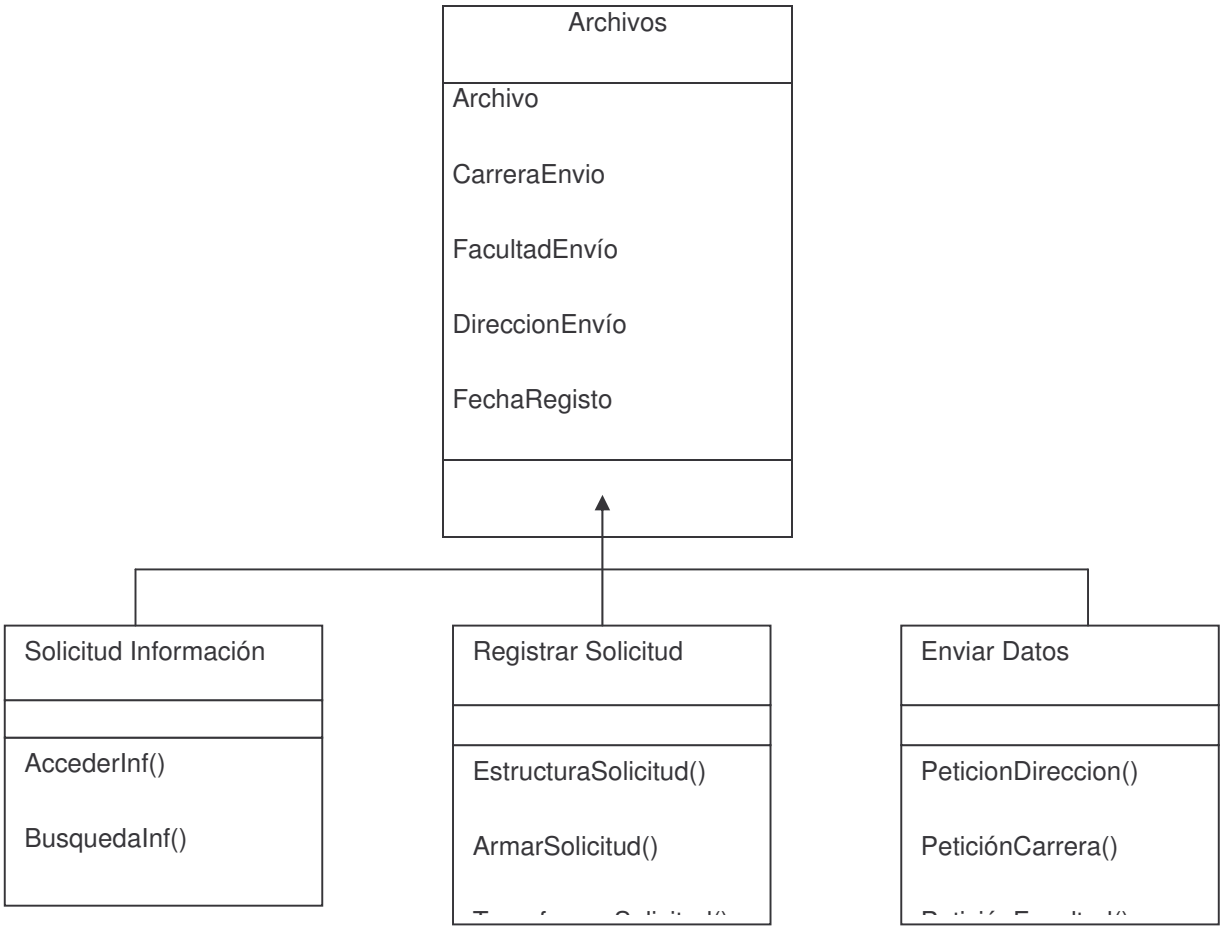


Figura 3.8: Diagrama de Clases Secundarias

Como se puede observar en la **figura 3.8** esta clase tiene sus clases secundarias es decir que los proceso que se muestran en las clases secundarias dependen de la clase archivo.

La siguiente clase es la de respuesta que tiene el sistema que le envío el archivo:

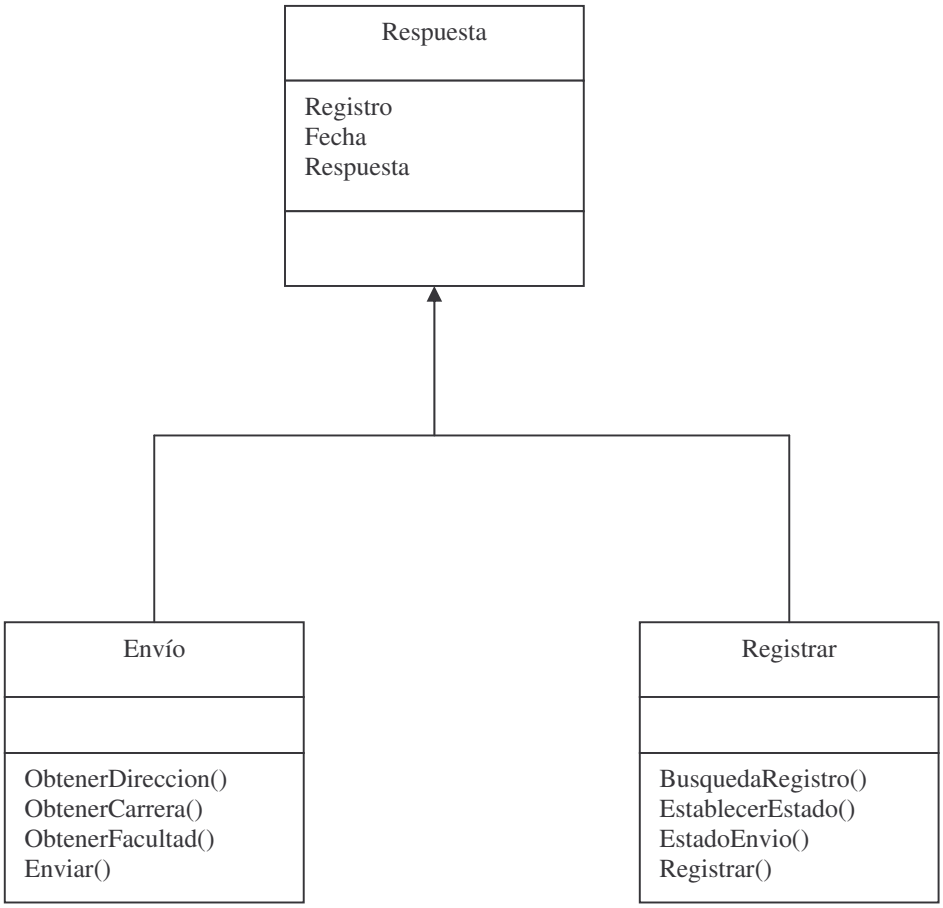
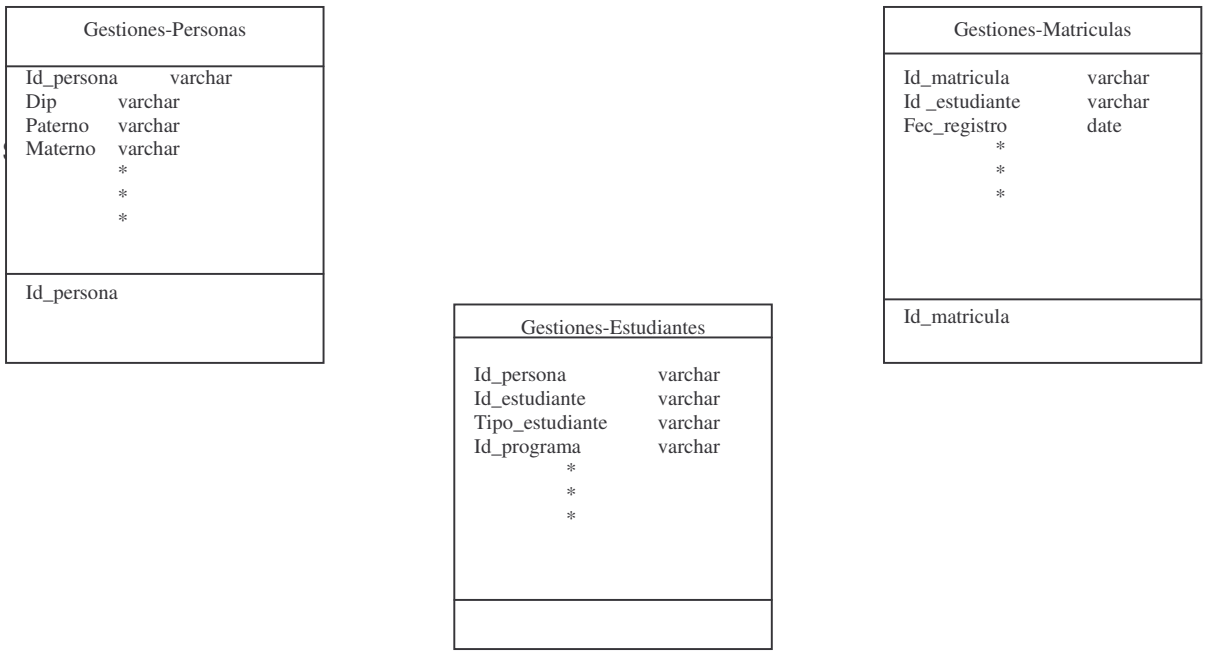


Figura 3.9: Diagrama de Clases de Respuesta

3.7 MODELO ENTIDAD RELACIÓN

A continuación mostraremos las tablas ya implementadas en los sistemas tanto de Gestiones como el Académico, estas tablas son con las que interactúa la aplicación.



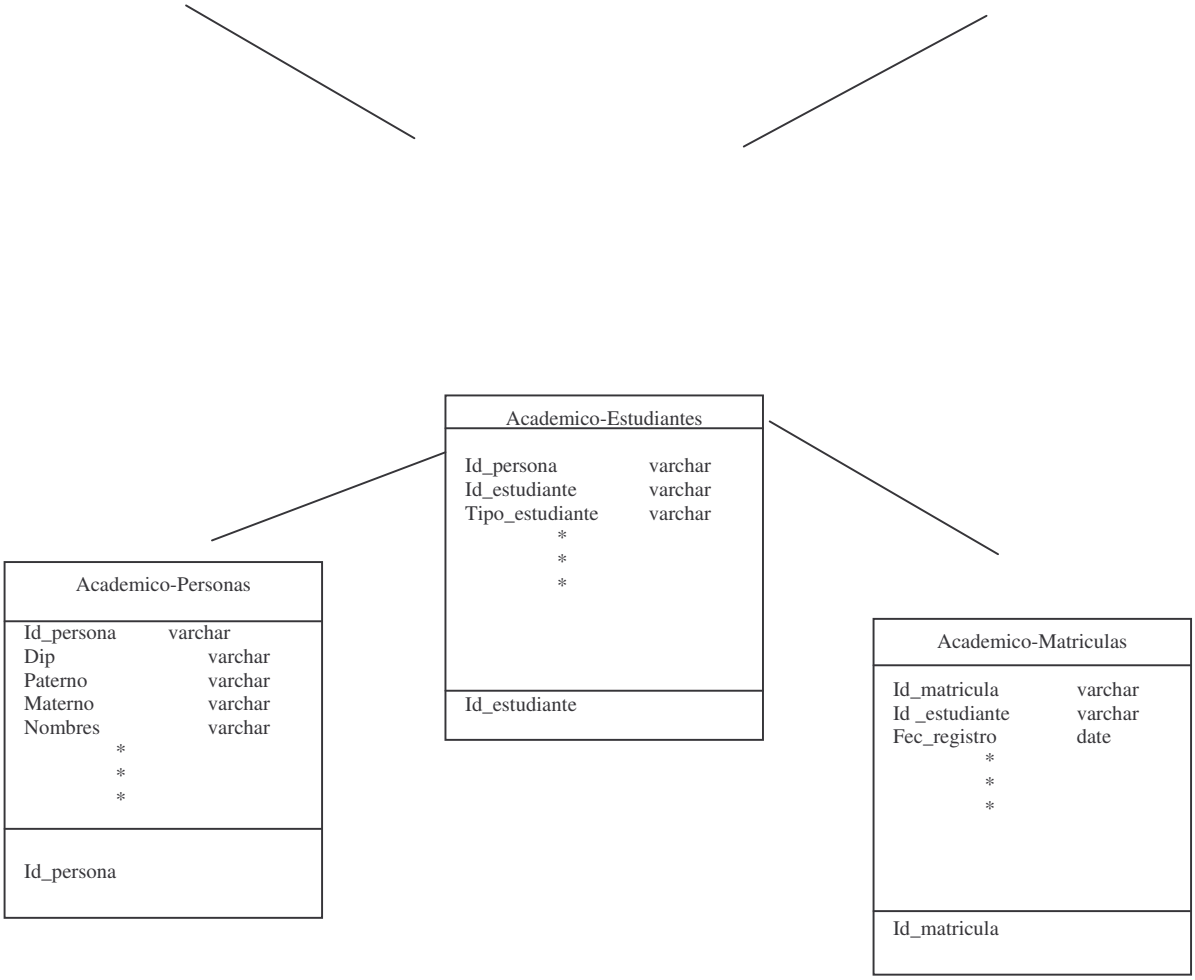


Figura 3.10: Diagrama Entidad Relación

3.8 DISEÑO Y DESARROLLO DE LA BASE DE DATOS

Ahora vamos a pasar a la descripción de las tablas con las que se relaciona nuestra aplicación

Personas.- Esta entidad representa a todas las personas que intervienen en el sistema de Gestiones como el Académico. La tabla contiene los datos propios de cada persona.

Nombre	Tipo	Descripción
Id_persona	Varchar	Identificador único del administrador
Dip	Varchar	Número de Documento personal
Paterno	Varchar	Apellido paterno de la persona
Materno	Varchar	Apellido materno de la persona
Nombres	Varchar	Nombres de la persona

Tabla 3.1: Descripción de la tabla de personas

Estudiantes.- Esta tabla es una lista de los estudiantes que son parte de la UMSA, en esta tabla se registran los estudiantes nuevos.

Nombre	Tipo	Descripción
Id_persona	Varchar	Identificador de la persona
Id_estudiante	Varchar	Es el registro universitario
Fec_registro	Date	Fecha de creación
Fec_modificacion	Date	Fecha de la modificación
Ult_usuario	Varchar	El usuario que almaceno el registro

Tabla 3.2: Descripción de la tabla Estudiantes

Matriculas.- La tabla contiene un histórico de las matriculas de cada uno de los estudiantes.

Nombre	Tipo	Descripción
Id_matricula	Varchar	El numero de la matricula
Id_estudiante	Varchar	Registro universitario del estudiante

Fec_registro	Date	Fecha de creación
Fec_modificacion	Date	Fecha de la modificación
Ult_usuario	Varchar	El usuario que almaceno el registro
Estado	Varchar	Estado del registro

Tabla 3.3: Descripción de la tabla Matriculas

3.9 DESARROLLO DE LA APLICACION

Una ves construidas nuestras Bases de Datos, debemos programar nuestra aplicación, es decir la interface que nos permitirá realizar el intercambio de información entre nuestros sistemas que se encuentran distribuidos. Para el desarrollo de nuestra aplicación haremos uso de algunas herramientas, como el XML, SAX, DOM y JAVAMAIL

Desarrollaremos scripts que nos permitirán interpretar y extraer la información, para luego ser enviada a su receptor. La siguiente figura muestra los procesos que debe seguir para el envió de información.

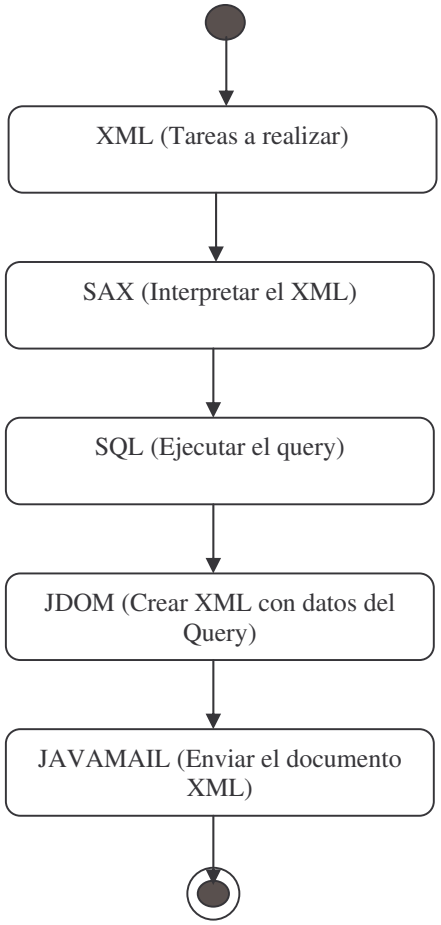


Figura 3.11: Diagrama de Actividades para el Envió de Información

Como podemos apreciar en la figura anterior, partiremos de un XML que tendrá la información que queremos enviar al sistema receptor. Para leer el XML utilizaremos el **SAX**, el SAX se utiliza para hacer un recorrido secuencial de los elementos del documento XML.

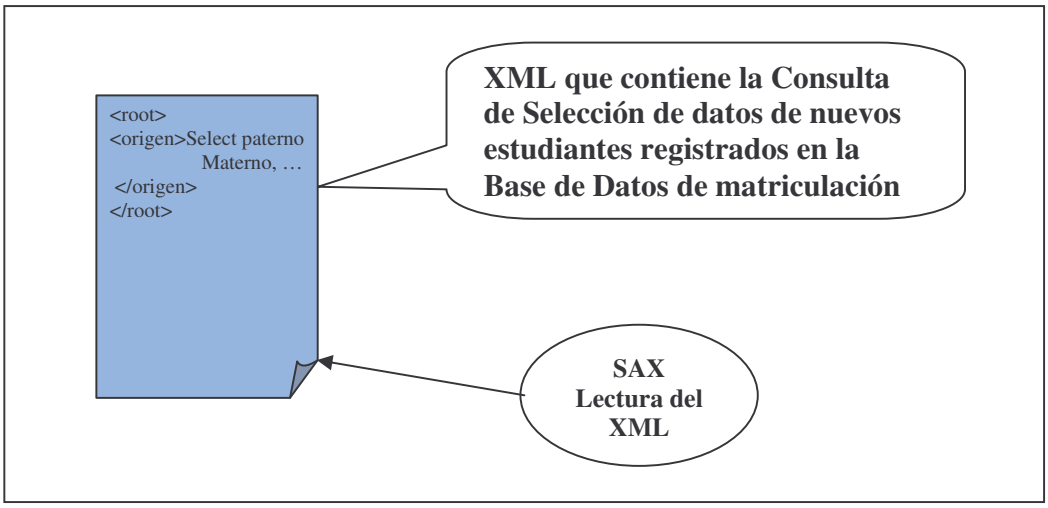


Figura 3.12: Representa el XML y la lectura mediante el SAX

Una ves que se hizo la lectura del XML, utilizando el manejoSQL se ejecuta el Query (Consulta de SQL), esta ejecución de la consulta nos proporciona una lista con los datos que solicitaron en la consulta. Seguidamente realizamos la creación del documento XML con los datos obtenidos del Query (consulta), esta creación del documento se hace con la utilización del JDOM y con este proceso tendríamos los datos empaquetados y listos para ser enviados como se muestra en la figura 3.13, debemos hacer notar que al crear el documento XML cada atributo de la consulta se convierte en una etiqueta del XML, entonces si el resultado de la consulta tuviese diez registros, cada etiqueta se repetirá diez veces con diferentes datos en su contenido de la etiqueta.

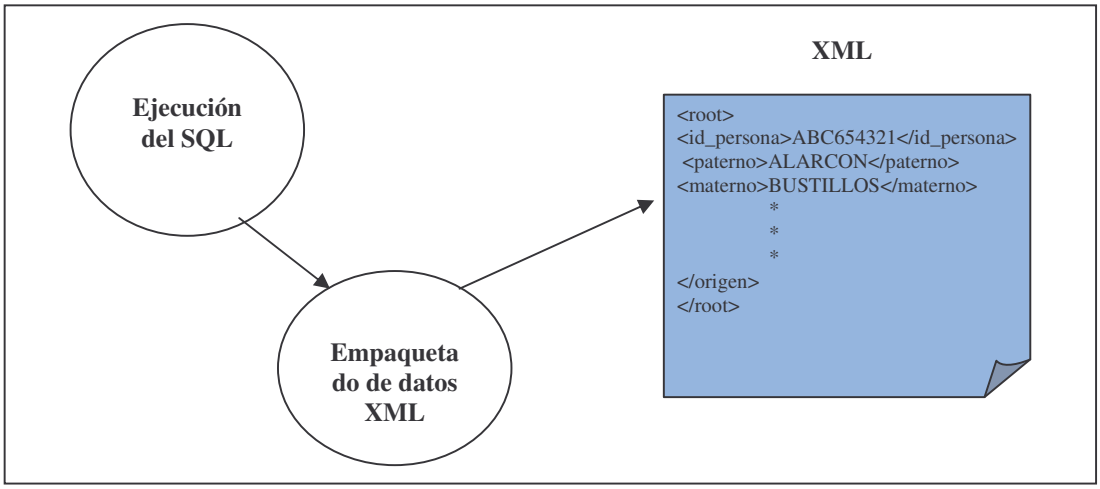


Figura 3.13: Empaquetado de la información en XML

Después de que fue empaquetada la información dentro de un XML esta listo para el envió del mismo y lo hacemos a un servidor SMTP de correo, este servidor se encarga de la administración de este documento XML.

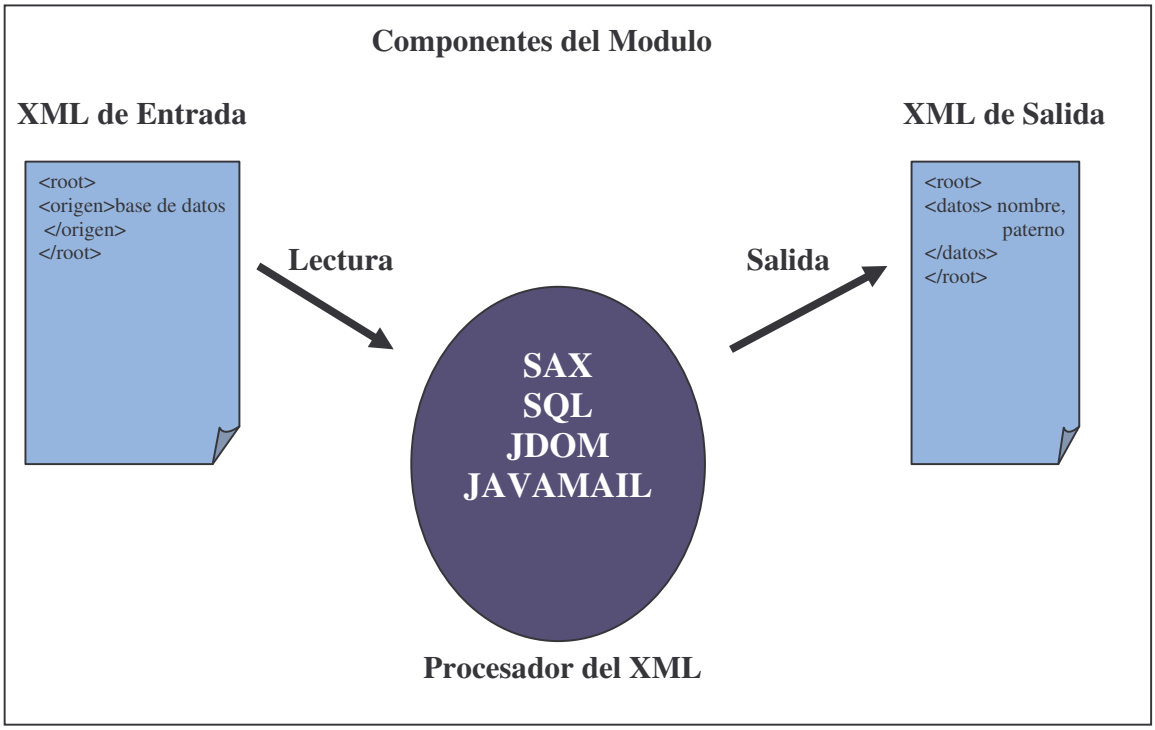


Figura 3.14: Empaquetado de la información y envió con el JavaMail

El servidor de correo hace llegar los datos al destino correcto, entonces nuestra aplicación en el sistema académico se encarga de desempaquetar la información y almacena en su Base de Datos respectiva.

El almacenamiento lo hace con un XML adicional, este otro XML es el que se encarga de decirle a la aplicación en que tablas debe almacenar la información proveniente del otro sistema, es decir que no interesa como vengan los datos en el XML que nos enviaron, el administrador del

sistema académico es el que le da la forma de almacenamiento a través del otro XML, veamos la figura siguiente

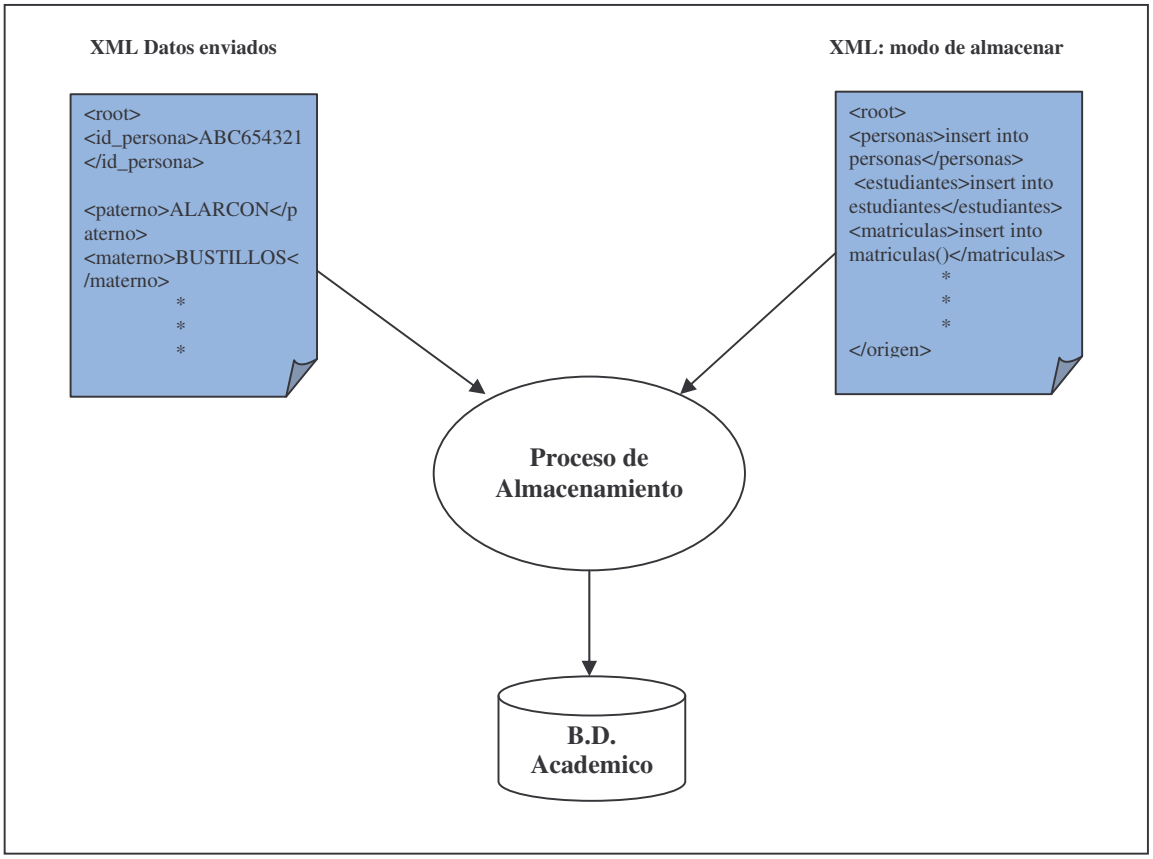


Figura 3.15: Almacenamiento de la Información

Entonces si nosotros queremos realizar acciones con nuestros datos escritos en XML tenemos dos mecanismos para acceder a documentos XML y trabajar con ellos. Se tratan simplemente de unas normas que indican a los desarrolladores la manera de acceder a los documentos. Estas normas incluyen una jerarquía de objetos que tienen métodos y atributos con

los que tendremos que trabajar y que nos simplificarán las tareas relativas al recorrido y acceso a las partes del documento

Estos dos mecanismos se denominan **SAX y DOM**. SAX se utiliza para hacer un recorrido secuencial de los elementos del documento XLM y DOM implica la creación de un árbol en memoria que contiene el documento XML, y con él en memoria podemos hacer cualquier tipo de recorrido y acciones con los elementos que queremos.

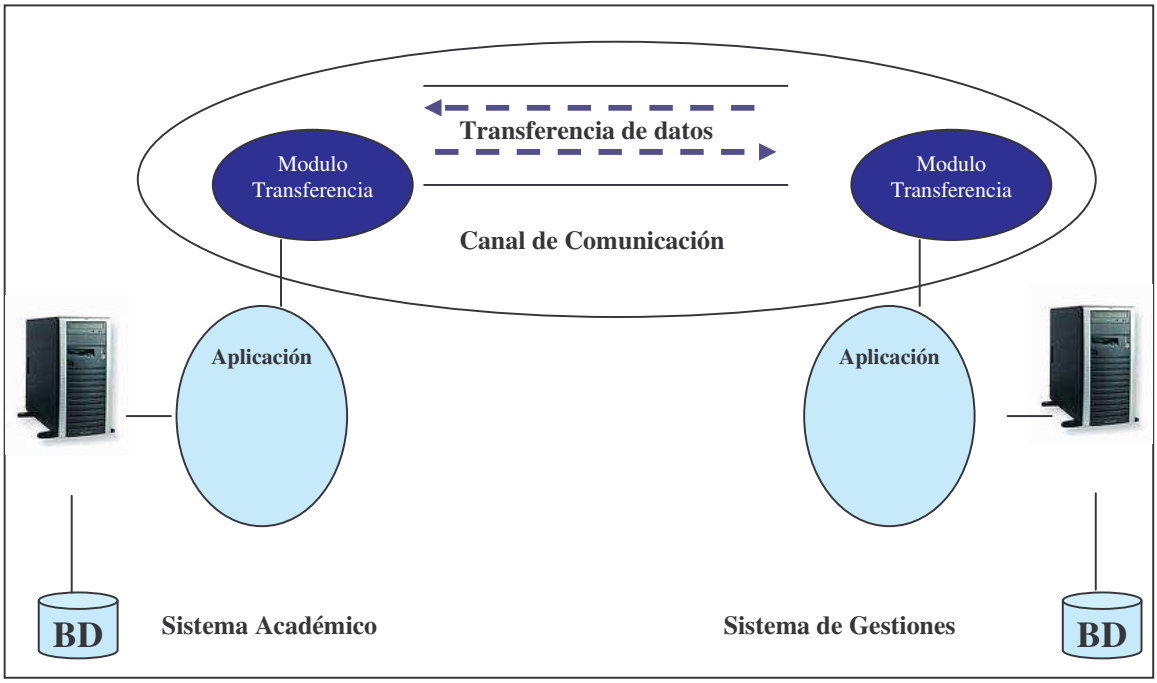


Figura 3.12: Diagrama de Transformación, envío y recepción de datos

3.10 MODELO PARA LA ESTIMACIÓN DE CALIDAD DEL WEB SERVICE

La calidad de un Web Service es vital para una organización que busca el mejor rendimiento de sus sistemas. Se propone Un Modelo de Especificación de Calidad para Web Services, basado en características, con un total de 46 métricas.

El principal aporte de este modelo es que parte de una base ya probada como es (MOSCA), el cual a su ves se inspira en estándares de calidad y propone métricas, para luego incorporar los aspectos propios de la calidad de los Web Services y Calidad de Software, donde este ultimo es una fuente de requerimientos importante íntimamente relacionados al mundo de la ingeniería Web y las aplicaciones en Internet.

En este modelo se evalúan seis categorías que son: Funcionalidad, Eficiencia, Mantenibilidad, Fiabilidad, Usabilidad y Portabilidad, pero la Usabilidad se le da poca importancia por ser los Web Services una capa Middleware con la que el usuario no actúa directamente. Ahora pasamos a describir las categorías:

Categorías	**Definición**
Funcionalidad (FUN)	Es la capacidad del Webservice para proveer las funciones que cumplan con las necesidades específicas o implícitas, cuando es utilizado bajo ciertas condiciones.
Fiabilidad (FIA)	La fiabilidad es la capacidad del Webservice para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones especificadas.
Usabilidad (USA)	Esta categoría se refiere a la capacidad del Webservice para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones específicas.
Eficiencia (EFI)	Es la capacidad del Webservice para proveer un rendimiento apropiado, relativo a la cantidad de recursos utilizados, bajo condiciones específicas.
Mantenibilidad	Es la capacidad del Webservice para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software ante cambios del ambiente, en requerimientos y especificaciones funcionales.

(MAB)	
Portabilidad (POR)	La portabilidad es la capacidad del Webservice para ser transferido de un ambiente a otro.

Tabla 3.7 Categorías del Modelo de Calidad de Webservices.

La siguiente tabla muestra los rangos utilizados en las diferentes escalas:

Escala	*Rango*
1	Respuestas correctas / Respuestas procesadas
2	Actividades ejecutadas completamente / Actividades totales
3	5 = Si 1 = No
4	5= Siempre, 4= Casi siempre, 3= En ocasiones, 2= Muy poco, 1= Nunca
5	Cantidad de transacciones exitosas / Cantidad de transacciones totales.
6	Funcionalidades que pertenecen a otros sistemas / Funcionalidades del WS
7	Tiempo de respuesta real / Tiempo de respuesta requerido
8	Cant. de datos / 60 seg
9	5 = menos de 15 seg, 4 = entre 16 seg y 30 seg, 3 = entre 31 seg y 1 min, 2 = entre 1 min y 5 min, 1 = más de 5 min
10	Cantidad de peticiones atendidas / número de peticiones
11	5 = menos de 1 min, 4 = entre 1 min y 10 min, 3 = entre 10 min y 20 min, 2 = entre 20 min y 1 hora, 1 = más de 1 hora
12	Cantidad de solicitudes que atiende actualmente / cantidad de solicitudes que es capaz de atender
13	Cantidad de solicitudes que es capaz de atender actualmente / cantidad de solicitudes que es capaz de atender en un año
14	4=Ninguna, 3=Muy pocas, 2=Algunas, 1=Muchas
15	5=Semestral o más, 4=Trimestral, 3= Mensual, 2= Semanal, 1= Diario
16	Cantidad de fallas resueltas / Cantidad de fallas ocurridas
17	5=Muy fácil, 4=Fácil, 3=Promedio, 2=Difícil, 1= Muy difícil
18	5 = Muy baja, 4= Baja, 3 = Media, 2 = Alta, 1 = Muy alta
19	5 = 5 SO, 4 = 4 SO, 3 = 3 SO, 2 = 2 SO, 1 = 1 SO

20	N° de productos requeridos con los que co-existe / N° de productos requeridos
21	5 =Totalmente, 4 = Casi todo, 3 = Medianamente, 2 = Poco, 1 =Ninguno

Tabla 3.8 Tipos de escala para las métricas.

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUCIONES

El resultado del proyecto descrito en este trabajo es la implantación de una herramienta para la integración de dos sistemas, en este caso el Sistema de Gestiones de la UMSA con el Sistema Académico de Odontología, pues este permitirá la actualización de la información en forma automática entre ambos sistemas.

Debemos hacer notar que el desarrollo de este proyecto no solo sirve para el intercambio de información de estos dos sistemas, se puede utilizar también para el intercambio de información entre el Sistema de Gestiones y el Sistema Académico de otras carreras o facultades de la UMSA siempre y cuando estén instalados los sistemas del UMSATIC, en estos momentos es muy necesario para tener información en línea.

Para el desarrollo del proyecto, fue muy importante la adopción de una metodología orientada a objetos, ya que permitió modelar los sistemas y la interacción entre ellos de una manera mucho mas visible y cercana al mundo real.

De la misma manera permitió modelar os casos de uso tomando en cuenta la dificultad y la urgencia de cada uno de ellos, de esta manera se logro un desarrollo más rápido y adaptado a las necesidades del proyecto.

Durante el desarrollo del proyecto, se observo que la adaptación del modulo que permite la actualización de la información de lo sistemas distribuidos es bastante complicada, entonces es necesario un estudio a fondo y detallado para identificar donde tiene que incluirse este modulo.

Debe existir una buena documentación del modulo de intercambio de información, en caso contrario esta implentación será muy dura y compleja porque se tendrá que ir directo al estudio de la implentación para entender que fue lo que se desarrollo.

Finalmente con la documentación del modulo de intercambio de información que se presenta en este trabajo, se desea que las actualizaciones entre lo sistemas sea mas sencillo posible.

4.2 RECOMENDACIONES

Cumplidos los objetivos del Proyecto de Grado, y en aras de mejoramiento del proyecto de investigación, es posible dar algunas recomendaciones en busca del crecimiento del mismo.

El avance acelerado de la tecnología, así como las constantes apariciones de nuevas versiones herramientas de desarrollo que buscan eliminar errores en las versiones anteriores o mejorar, deben ser tomados en cuenta para el desarrollo del proyecto, ya que este debería avanzar al mismo ritmo que avanzan las herramientas de desarrollo, para aprovechar al máximo las nuevas funcionalidades y la corrección de errores en versiones anteriores.

A manera de aporte de investigación y al crecimiento del modulo de intercambio de información, se puede señalar que se debería estudiar la manera de enviar información en tiempo real a otros sistemas sin que el operador del sistema tenga que ejecutar algún evento extra al sistema. Esto opción daría a los usuarios una carga menos en sus tareas como operadores del sistema ya sea de gestiones o sistema académico.

GLOSARIO

A

Aplicaciones: Son los programas de computación implementados en un lenguaje para ser ejecutados y realizar una tarea.

Arquitectura: Se refiere al diseño de un sistema a un software, hardware o una combinación de ambos. Se utiliza para definir un sistema en líneas generales.

B

Backup: Es un respaldo de seguridad. Generalmente se refiere a respaldo de datos o información.

Base de Datos: Una colección de información organizada de tal manera que puede ser manejada fácilmente. Generalmente se refiere a base de datos computacionales y son manejadas por software.

C

Cifrar: Transcribir en guarismos, letras o símbolos, de acuerdo con una clave, un mensaje cuyo contenido se quiere ocultar.

Clase de Java: Es una pieza de software que define datos (el estado) y métodos (el comportamiento) de un objeto en Java.
CORBA: Una plataforma que permite que aplicaciones basadas en el paradigma de Objetos (o componentes) distribuidos se comuniquen entre si sin importar dónde se encuentren ni como fueron diseñadas.

Componente: Término genético que se aplica a los componentes no físicos de un sistema informático, por ejemplo los programas, sistemas operativos, etc. que permiten a este ejecutar sus tareas.

E

Estándar: Sirve como tipo, modelo, norma, patrón o referencia por ser corriente de serie.

Ethernet: Es una tecnología para redes de computadores basada en la transferencia de información de paquetes de tamaño variable, a una velocidad de transmisión de 10Mb/s. Las especificaciones de la capa física y capas de software de bajo nivel pueden ser encontradas en el estándar 802.3 de la IEEE.

F

Fast Ethernet: Es una tecnología de redes basada en el estándar de “Ethernet”. Soporta una velocidad de transmisión de 100 Mb/s. Las especificaciones están en el estándar 802.3u de la IEEE.

H

Heterogéneo: Compuesto de componentes o partes de distinta naturaleza.

L

Librería: Es un conjunto de rutinas precompiladas para ser utilizadas por los programas. Muchas veces es llamada Módulo.

M

Metalenguaje: Lenguaje natural o formal que se usa para explicar o hablar del lenguaje mismo o de una lengua. Las gramáticas formales son metalenguajes.

Metasistema: Es un conjunto de recursos distribuidos heterogéneos, de simple acceso, que se usan como un computador único. Es sinónimo de GRID Computacional.

MPI: (Message Passing Interface) es un estándar para la creación de librerías que permitan el pase de menajes entre programas.

O

OMG: Object Management Group, es una agrupación internacional sin fines de lucro que provee estándares para el desarrollo de componentes de software distribuido, interoperable, escalable y reusable.

Orientación a Objetos: Es un paradigma. Un sistema es orientado a objetos si maneja principalmente a diferentes tipos de objetos y las acciones que se pueden hacer dependen del tipo de objeto que se esté manejando.

P

Proceso: Un programa en ejecución.

Protocolo: Es un acuerdo en el formato para la transmisión de datos entre dos componentes.

R

Red de Computadoras: Es un conjunto de computadores y dispositivos conectados entre sí para intercambiar información y compartir recursos.

S

Sistema de Archivos: es un componente importante de un sistema de operación y suele contener: métodos de acceso, administración de archivos, administración del almacenamiento auxiliar e integridad de archivos. Está relacionado especialmente
con la administración del espacio de almacenamiento secundario, fundamentalmente con el almacenamiento de disco.

Sistema de Operación: Es el programa principal que es ejecuta en cualquier computadora. Se encarga de controlar los dispositivos periféricos, los sistemas de archivos, memoria y ejecución de procesos.

ANEXOS

ANEXO A: CARACTERISTICAS DEL J2EE [12]

J2EE es una plataforma creada por Sun en el año 1997, para el desarrollo de aplicaciones distribuidas orientadas principalmente a la empresa. En el desarrollo empresarial se ponen de manifiesto ciertos requisitos esenciales, no tan críticos en otras aplicaciones, que debe cumplir un desarrollo y que con J2EE podemos conseguir utilizando principalmente software libre, entre estos requisitos se pueden nombrar los siguientes:

- Escalabilidad, tanto horizontal como vertical.
- Fiabilidad.
- Facilidad de mantenimiento.
- Seguridad.
- Rendimiento.

- Extensibilidad.

- Flexibilidad

El objetivo final será conseguir la máxima productividad de los desarrolladores, por ello el uso de herramientas que en esta plataforma nos permitan un desarrollo ágil resultara relevante para el triunfo de nuestro producto.
El uso de J2EE nos ofrece múltiples ventajas:
Es una especificación que se puede utilizar en distintas plataformas.

Control por JCP, es un conjunto de grandes empresas que se encarga de la correcta evolución de la plataforma, entre ellas podemos incluir: Sun, IBM, Oracle, HP, etc.

Soluciones libres, ya que existen numerosos frameworks, y Apis Open Source para el desarrollo en este entorno. Asegurar la competencia, con productos de distintos precios y calidad. No obstante, no son todo ventajas, también existen una serie de inconvenientes:

Dependencia de un único lenguaje.

Complejidad que dificulta su adopción por los desarrolladores menos experimentados.

Muchos modelos de desarrollo, frameworks y Apis que pueden confundir a nuestros desarrolladores.

En concreto, podemos dividir la definición de J2EE en dos partes:

JSR: Java Specification Request, sirve para solicitar que una tecnología sea incluida en la especificación de J2EE, se presenta la propuesta y queda a expensas de la aprobación, si se aprueba debe quedar totalmente definida y el equipo encargado de su desarrollo además debe proporcionar un test de compatibilidad y una implementación de referencia.

JCP: Java Community Process, Conjunto de empresas formando un organismo que se encargan del control de J2EE, de esta manera se evita que pueda ser criticada por pertenecer sólo a Sun.

La forma de entrar es a través de una cuota anual, no obstante presenta diferentes restricciones desde el punto de vista del software libre (patentes, licencias propietarias y potestades superiores de Sun), a lo largo de distintas versiones fueron corregidas gracias principalmente al impulso de Apache.

En conclusión, desde el punto de vista teórico J2EE es una especificación de especificaciones para distintos conceptos: XML, Servicios Web, etc. También contamos con un conjunto de buenas prácticas o guías que se denominan J2EE Blueprints.

Una vez que ya hemos establecido teóricamente la definición de J2EE, empezaremos definiendo el modelo de capas que propone, con la posibilidad de variación según la complejidad y las necesidades que tengamos, de todas formas generalizando podemos realizar la siguiente separación de capas:

Cliente: Aquí se sitúan los distintos clientes de nuestra aplicación, normalmente un interfaz de usuario.

Presentación: Contiene la lógica de interacción entre usuario y aplicación. Controla la interacción entre usuario y lógica de negocio utilizando distintas vistas.

Lógica de Negocio: Código que realiza las funcionalidades que ofrece nuestra aplicación, aquí es dónde se pone de manifiesto la necesidad de fácil mantenimiento y extensibilidad.

Integración: Comunicación con otros subsistemas, como motores de bases de datos, de reglas, etc. Es importante la necesidad de que en esta capa se puedan añadir nuevas fuentes con cierta facilidad. Deberemos garantizar acceso transparente a las fuentes de información, con el uso de por ejemplo el patrón de diseño DAO (Data Access Object).

Sistemas de información: Son las fuentes de información: bases de datos, ficheros, etc. [11]

ANEXO B. HERRAMIENTAS AUXILIARES EN EL DESARROLLO DEL MODULO DE TRANSACCIONES [13]

XML

XML, es el estándar de Extensible Markup Language. XML no es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes.

XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.[9]

En primer lugar para entenderlo bien hay que olvidarse un poco, sólo un poco de HTML. En teoría HTML es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que XML es un subconjunto de SGML especializado en la gestión de información para la Web.

En la práctica XML contiene a HTML aunque no en su totalidad. La definición de HTML contenido totalmente dentro de XML y por lo tanto que cumple a rajatabla la especificación SGML es XHTML (Extensible, Hypertext Markup Language). [9]

Desde su creación, XML ha despertado encontradas pasiones, y como para cualquier tema en Internet, hay gente que desde el principio se deja iluminar por sus expectativas, mientras otras muchas lo han ignorado.

Historía y Objetivos del XML

XML fue creado al amparo del Word Wide Web Consortium (W3C) organismo que vela por el desarrollo de WWW partiendo de las amplias especificaciones de SGML.

Su desarrollo se comenzó en 1996 y la primera versión salió a la luz el 10 de febrero de 1998. La primera definición que apareció fue: *Sistema para definir validar y compartir formatos de documentos en la Web*.

Durante el año 1998 XML tuvo un crecimiento exponencial, y con ello me refiero a sus apariciones en medios de comunicación, menciones en páginas Web, soporte software, etc.

Respecto a sus objetivos son:

- XML debe ser directamente utilizable sobre Internet.
- XML debe soportar una amplia variedad de aplicaciones.
- XML debe ser compatible con SGML.
- Debe ser fácil la escritura de programas que procesen documentos XML.
- El número de características opcionales en XML debe ser absolutamente mínima, idealmente cero.
- Los documentos XML deben ser legibles por humanos y razonablemente claros.
- El diseño de XML debe ser preparado rápidamente.
- El diseño de XML debe ser formal y conciso.
- Los documentos XML deben ser fácilmente creables.

- La concisión en las marcas XML es de mínima importancia.

Esta especificación, junto con los estándares asociados (Unicode e ISO/IEC 10646 para caracteres, Internet RFC 1766 para identificación de lenguajes, ISO 639 para códigos de nombres de lenguajes, e ISO 3166 para códigos de nombres de países), proporciona toda la información necesaria para entender la Versión 1.0 de XML y construir programas de computador que los procesen.[9]

Estructura del XML

El *metalenguaje* XML consta de cuatro especificaciones (el propio XML sienta las bases sintácticas y el alcance de su implementación):

DTD (*Document Type Definition*): Definición del tipo de documento. Es, en general, un archivo/s que encierra una definición formal de un tipo de documento y , a la vez, especifica la estructura lógica de cada documento. Define tanto los elementos de una página como sus atributos. El DTD del XML es opcional. En tareas sencillas no es necesario construir una DTD, entonces se trataría de un documento "bien formado"(*well-formed*) y si lleva DTD será un documento "validado" (*valid*).

XSL (*eXtensible Stylesheet Language*): Define o implementa el lenguaje de estilo de los documentos escritos para XML. Desde el verano de 1997 varias empresas informáticas como Arbortext, Microsoft e Inso vienen trabajando en una propuesta de XSL (antes llamado "xml-style") que presentaron a W3C. Permite modificar el aspecto de un documento. Se puede lograr múltiple columnas, texto girado, orden de visualización de los datos de una tabla, múltiples tipos de letra con amplia variedad en los tamaños.

Este estándar está basado en el lenguaje de semántica y especificación de estilo de documento (DSSSL, *Document Style Semantics and Specification Language*, ISO/IEC 10179) y, por otro lado, se considera más potente que las hojas de estilo en cascada (CSS, *Cascading Style Sheets*), usado en un principio con el lenguaje DHTML.

"Se espera que el CSS sea usado para visualizar simples estructuras de documentos XML (actualmente se ha conseguido mayor integración en XML con el protocolo CSS2 (*Cascading Style Sheets, level 2*) ofreciendo nuevas formas de composición y una más rápida visualización) y, por otra parte, XSL pueda ser utilizado donde se requiera más potencia de diseño como documentos XML que encierran datos estructurados (tablas, organigramas, etc.).

XLL (*eXtensible Linking Language*): Define el modo de enlace entre diferentes enlaces. Se considera que es un subconjunto de HyTime (*Hipermedia/Timed-based structuring Language* o Lenguaje de estructuración hipermedia/basado en el tiempo, ISO 10744) y sigue algunas especificaciones del TEI (*Text Encoding Initiative* o Iniciativa de codificación de texto). Desde marzo de 1998 el W3C trabajo en los enlaces y direccionamientos del XML. Provisionalmente se le renombró como *Xlink* y a partir de junio se le denomina XLL. Este lenguaje de enlaces extensible tiene dos importantes componentes: *Xlink* y el *Xpointer*. Va más allá de los enlaces simples que sólo soporta el HTML. Se podrá implementar con enlaces extendidos. Jon Bosak establece los siguientes mecanismos hipertextuales que soportará esta especificación:

- Denominación independiente de la ubicación.

- Enlaces que pueden ser también bidirecccionales.

- Enlaces que pueden especificarse y gestionarse desde fuera del documento a los que se apliquen (Esto permitirá crear en un entorno intranet/extranet un banco de datos de enlaces en los que se puede gestionar y actualizar automaticamente. No habrá más errores del tipo "404 Not Found").

- Hiperenlaces múltiples (anillos, múltiples ventanas, etc.).

- Enlaces agrupados (múltiples orígenes).

- Transclusión (el documento destino al que apunta el enlace aparece como parte integrante del documento origen del enlace).

- Se pueden aplicar atributos a los enlaces (tipos de enlaces).

XUA (*XML User Agent*): Estandarización de navegadores XML. Todavía está en proceso de creación de borradores de trabajo. Se aplicará a los navegadores para que compartan todos las especificaciones XML.[9]

ANEXO B.1: DOM y SAX [14]

Han sido desarrollados dos métodos de analizar sintácticamente un documento XML. En el primer método, DOM (Document Object Model), se lee el documento completo y se identifica su estructura jerárquica.

El segundo método, SAX (Standard API for XML), consiste en ir identificando las marcas a medida que se va leyendo el documento. El segundo método es obviamente más rápido y consume menos recursos, pero tiene la desventaja de que cada vez que aparece una marca se debe decidir que hacer con ella, y no se puede regresar para atrás en el documento.

SAX ha sido desarrollado con aplicaciones de servidor en mente; el servidor debe suministrar rápidamente el resultado de transformar un documento XML. DOM fue desarrollado con aplicaciones de cliente en mente; por ejemplo un editor de XML necesita poder navegar en cualquier dirección la estructura del documento; en este caso el método SAX no sería muy útil.

Existen varias librerías disponibles que implementan un u otro método en varios lenguajes de programación diferentes. Veamos un ejemplo de un programa perl que usa el módulo XML::Dom para sacar información de un fichero XML:

Se puede programar con el lenguaje de programación que se desee para acceder a un documento XML. Los creadores del lenguaje son los responsables de crear unas API que cumplan las especificaciones de XML para que luego los desarrolladores de cada lenguaje las encuentren y puedan trabajar con ellas. Un lenguaje típico para trabajar con XML es Java y en este caso es SUN Microsystems la encargada de proveer el API y por lo tanto, los desarrolladores en Java cuentan con unas clases especiales que ha creado SUN para programar con XML.

Por su parte, los creadores de algunos lenguajes han implementado una tercera manera de programar con XML que se llama XSLT. Empresas como por ejemplo la organización Apache, SUN o Microsoft, ya la están apoyando, aunque en el W3C no han dicho que sea un estándar.

El trabajo con bases de datos y XML se está desarrollando con un lenguaje que se llama XQL (XML Query Language), que es uno de los ejemplos de lenguaje que sólo está publicado en el W3C como una "nota". [12]

ANEXO B.2: PROTOCOLO

Es un conjunto de reglas y formatos que se utilizan para la comunicación entre procesos que realizan una determinada tarea. Se requieren dos partes: [5]

- **Especificación de secuencia de mensajes que se han de intercambiar.**

- **Especificación de formato de los datos en los mensajes.**

Un Protocolo permite que componentes heterogéneos de sistemas distribuidos puedan desarrollarse independientemente, y por medio de módulos de software que componen el protocolo, haya una comunicación transparente entre ambos componentes. Es conveniente mencionar que estos componentes del protocolo deben estar tanto en el receptor como en el emisor.

Ejemplos de protocolos usados en los sistemas distribuidos:

- IP: Protocolo de Internet.- **Protocolo de la capa de Red, que permite definir la unidad básica de transferencia de datos y se encarga del direccionamiento de la información, para que llegue a su destino de red.[5]**

- TCP: Protocolo de Control de Transmisión.- **Protocolo de la capa de transporte, que permite dividir y ordenar la información a transportar en paquetes de menor tamaño para su transporte y recepción.[5]**

- HTTP: Protocolo de Transferencia de Hipertexto.- **Protocolo de la capa de aplicación, que permite el servicio de transferencia de páginas de hipertexto entre cliente WEB y los servidores.[1]**

- SMTP: Protocolo de Transferencia de Correo Simple.- **Protocolo de la capa de aplicación, que permite el envío de correo electrónico por la red.[1]**

- POP3: Protocolo de Oficina de Correo.- **Protocolo de la capa de aplicación, que permite la gestión de correos en Internet, es decir, le permite a una estación de trabajo recuperar los correos que están almacenados en el servidor.[1]**

ANEXO C: CASOS DE USO COMPLEMENTARIOS

Caso de uso: < 3 > Solicitud de Información

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Obtiene la información solicitada.
Condición Final de Fallo: No obtener la información que solicitó.
Actor primario: Administrador de Gestiones.
Disparador: Solicitud de la información para el envío.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: El administrador selecciona la opción correspondiente para solicitar la información.
<PASO 2>: La información que se requiere es seleccionada de la base de datos.
<PASO 3>: La información solicitada esta almacenada en tablas temporales de acuerdo a la solicitud.

EXTENSIONES

<PASO 2a>: En caso de la información que se solicita no este en la bese de datos solo no envía ninguna solicitud.

Caso de uso: < 4 > Armar Solicitud

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Se arma la solicitud en un nuevo formato.
Condición Final de Fallo: No se arma la solicitud requerida.
Actor primario: Administrador de Gestiones.
Disparador: El administrador activa la solicitud de la información.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: La solicitud de la información es armada de acuerdo a los requerimientos del envío.
<PASO 2>: Se extrae el formato al que se quiere armar la información
<PASO 3>: El armado se realiza en el nuevo formato.

EXTENSIONES

<PASO 2a>: Pueden existir varios formatos de armado de la información se arma de acuerdo al requerimiento.

Caso de uso: < 5 > Transformar la Solicitud

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: La solicitud d la información es transformada al nuevo tipo de archivo.
Condición Final de Fallo: No se pudo transformar la información al nuevo tipo de archivo.
Actor primario: Administrador de Gestiones.
Disparador: Transformar la información para el envío.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: El administrador activa la transformación de la información al nuevo tipo de archivo.
<PASO 2>: La información es enviada para su transformación.

<PASO 3>: La información solicitada es transformada al nuevo tipo de archivo.

EXTENSIONES

<PASO 2a>: En caso de existir errores en la solicitud de la información no podrá ser enviada para u transformación al nuevo tipo de archivo.

Caso de uso: < 6 > Registrar la Solicitud Transformada

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Registrar la solicitud transformada en el nuevo tipo de archivo en la Base de Datos.
Condición Final de Fallo: No poder Registrar la solicitud en la Base de Datos.
Actor primario: Administrador de Gestiones.
Disparador: Registrar la solicitud en la Base de Datos.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: El administrador activa el grabado de la información transformada en la base de Datos.
<PASO 2>: La información es enviada para su grabado.

<PASO 3>: Verifica si no existe duplicidad en la Base de Datos de la información transformada.
<PASO 4>: Procede al grabado de la información dependiendo de la respuesta de la verificación.

EXTENSIONES

<PASO 3a>: En caso de existir duplicidad en la información este no se registra en la base de datos.

Caso de uso: < 7 > Enviar Datos Transformados

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Envío de información transformada para su grabado en otra base de datos que esta distribuido.
Condición Final de Fallo: No poder enviar la información.
Actor primario: Administrador de Gestiones.
Disparador: Activar en el evento envío de la información.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: El administrador activa el evento del envío de la información.
<PASO 2>: Realiza la consulta para obtener la información de la base de datos.
<PASO 3>: Obtiene la información que se solicita para el envío.

<PASO 4>: Envía la información al destinatario correspondiente.

EXTENSIONES

<PASO 2a>: En caso de no existir la información que se solicita no hace ningún listado.

<PASO 3a>: Si no existe información no lista como pendientes.

Caso de uso: < 8 > Registrar Información Enviada

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Registrar información enviada por otro sistema.

Condición Final de Fallo: No registrar información.

Actor primario: Administrador de Gestiones.

Disparador: Activar la recepción del archivo.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: Recibir la información.

<PASO 2>: Transformar la información de acuerdo a los requerimientos.

<PASO 3>: Registrar la información en las tablas que se requiere.

EXTENSIONES

<PASO 3a>: En caso de ser una nuevos datos registrar , s ya existe los datos actualizar en los campos requeridos de las tablas que este deba actualizar.

Caso de uso: < 9 > Enviar Respuesta

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Envío de respuesta de confirmación de haber recibido la información.
Condición Final de Fallo: No poder enviar respuesta de recibido.
Actor primario: Administrador de Gestiones.
Disparador: Activación automática del envío de respuesta.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: Después de haber grabado la información se habilita el envión de la respuesta.

<PASO 2>: Busca al el número IP al que lo envío la información para darle la respuesta.
<PASO 3>: Envía la respuesta conformidad al servidor que envío la información.

EXTENSIONES

<PASO 2a>: La búsqueda la realiza de una lista de números IP almacenado en la Base de Datos.

Caso de uso: < 10 > Almacenar Respuesta

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: Almacena la respuesta en la base de datos.
Condición Final de Fallo: No almacenar la respuesta.
Actor primario: Administrador de Gestiones.
Disparador: almacena la respuesta después del envío de esta.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: Después del envío de la respuesta recepciona dicha respuesta.
<PASO 2>: Realiza la búsqueda para ver a que envío pertenece la repuesta por un código interno de la base de datos.

<PASO 3>: Encuentra el registro de al que pertenece y cambia el estado del registro.

EXTENSIONES

<PASO 2a>: Debe encontrar en registro en forma obligatoria para su edición del mismo.
<PASO 3a>: cambia el estado para poner en estado pasivo el registro.

Caso de uso: < 11 > Iniciar la Sesión del Administrador del Sistema

---

INFORMACIÓN CARACTERÍSTICA

Condición Final de Éxito: El administrador debe poseer una cuenta de cliente del sistema.
Condición Final de Fallo: Se niega el acceso a los módulos del sistema .
Actor primario: Administrador de Gestiones.
Disparador: solicitud de entrada a los módulos del sistema.

ESCENARIO PRINCIPAL DE ÉXITO

<PASO 1>: Se muestran una ventana en donde se introducirán los datos correspondientes para acceder al sistema.

<PASO 2>: El usuario introduce el login y el Password.

<PASO 3>: El sistema valida los datos introducidos.

<PASO 4>: El sistema otorga acceso a los módulos del sistema.

EXTENSIONES

<PASO 3a>: El login o contraseña están o En caso de no existir la información que se solicita no hace ningún listado.

ANEXO D: MODELO DE CALIDAD PARA EL WEB SERVICES

Para el desarrollo del producto de calidad aquí propuesto, se partió de MOSCA. Este modelo fue desarrollado por el Laboratorio de Información y Sistemas de Información (LISI) de la Universidad Simón Bolívar, con el financiamiento del gobierno venezolano.

Se basa en un enfoque de calidad sistémica el cual contempla tanto la perspectiva producto (software) como la perspectiva proceso. Con base a esta experiencia y a su condición sistémica, se decidió hacer una adaptación del mismo para WS.

A continuación mostramos las características de cada categoría de evaluación:

Características	Descripción
FUN.1 Precisión	Capacidad del Webservice para proveer resultados o efectos correctos o convenientes. Esto incluye el grado de precisión de los valores calculados. Está relacionada con la correctitud en la ejecución de las transacciones.
FUN.2 Transición	Capacidad de tratar una secuencia de actividades como una simple unidad de trabajo.
FUN.3 Seguridad	Capacidad del Webservice de proteger su información y datos así como la de controlar el acceso no autorizado al mismo
FUN.4 Iteroperabilidad	Capacidad del Webservice para interactuar con uno o más sistemas (Propiciada por la tecnología – XML).

Tabla Anexo 1. Características del Modelo correspondiente a la categoría Funcionalidad

Características	Descripción
EFI.1 Estandarización	Capacidad del Webservice para ajustarse a estándares, convenciones o regulaciones.
EFI.2 Comportamiento del Tiempo	Capacidad del Webservice para proveer respuestas y tiempos de procesamiento apropiados en tiempo de ejecución bajo condiciones específicas.
EFI.3 Utilización de los recursos	Capacidad del Webservice para utilizar cantidades apropiadas de los recursos cuando el mismo ejecuta sus funciones bajo condiciones específicas.
EFI.4 Latencia	Tiempo transcurrido entre enviar una solicitud y recibir una respuesta.
EFI.5 Throughput	Capacidad del Webservice para representar el número de peticiones atendidas por un WS en un período de tiempo determinado.

Tabla Anexo 2. Características del Modelo correspondiente a la categoría Eficiencia.

Características	Descripción
FIA.1 Disponibilidad	Capacidad del producto del software de estar siempre presente y en estado operativo al momento de ejecutar una función en un período determinado, bajo condiciones específicas. Está relacionada con el tiempo de reparo.
FIA.2 Tiempo de reparo	Representa el tiempo que toma reparar un servicio que ha fallado.
FIA.3 Accesibilidad	Según QoS, representa la capacidad de un servicio para atender una solicitud de un WS. Puede ocurrir que un WS este disponible más no accesible. Está relacionada con sistemas altamente escalables
FIA.4 Escalabilidad	Habilidad de atender consistentemente las solicitudes a pesar de las variaciones del volumen de la demanda.
FIA.5 Tolerancia a fallas	Capacidad del producto de software para mantener un nivel de rendimiento específico en caso de errores en el software o de infracciones sobre sus interfaces.
FIA.6 Madurez	Capacidad del producto de software para evitar fallas como resultado de errores en el mismo.

Tabla Anexo 3. Características del Modelo correspondiente a la categoría Fiabilidad.

Características	Descripción
USA.1 Documentación	Este concepto está relacionado con la existencia de un documento WSDL donde se expone la funcionalidad, la accesibilidad y la comunicación del WS, entre otros.

Tabla Anexo 4. Características del Modelo correspondiente a la categoría Usabilidad.

Características	Descripción
MAB.1 Capacidad de análisis	Capacidad del producto de software para ser diagnosticado por deficiencias o fallas en el software.
MAB.2 Capacidad de cambio	Capacidad del producto de software para implementar una modificación específica de una manera más sencilla.
MAB.3 Estabilidad	Capacidad del producto de software para evitar efectos inesperados después de modificaciones en el software.

Tabla Anexo 5. Características del Modelo correspondiente a la categoría Mantenibilidad.

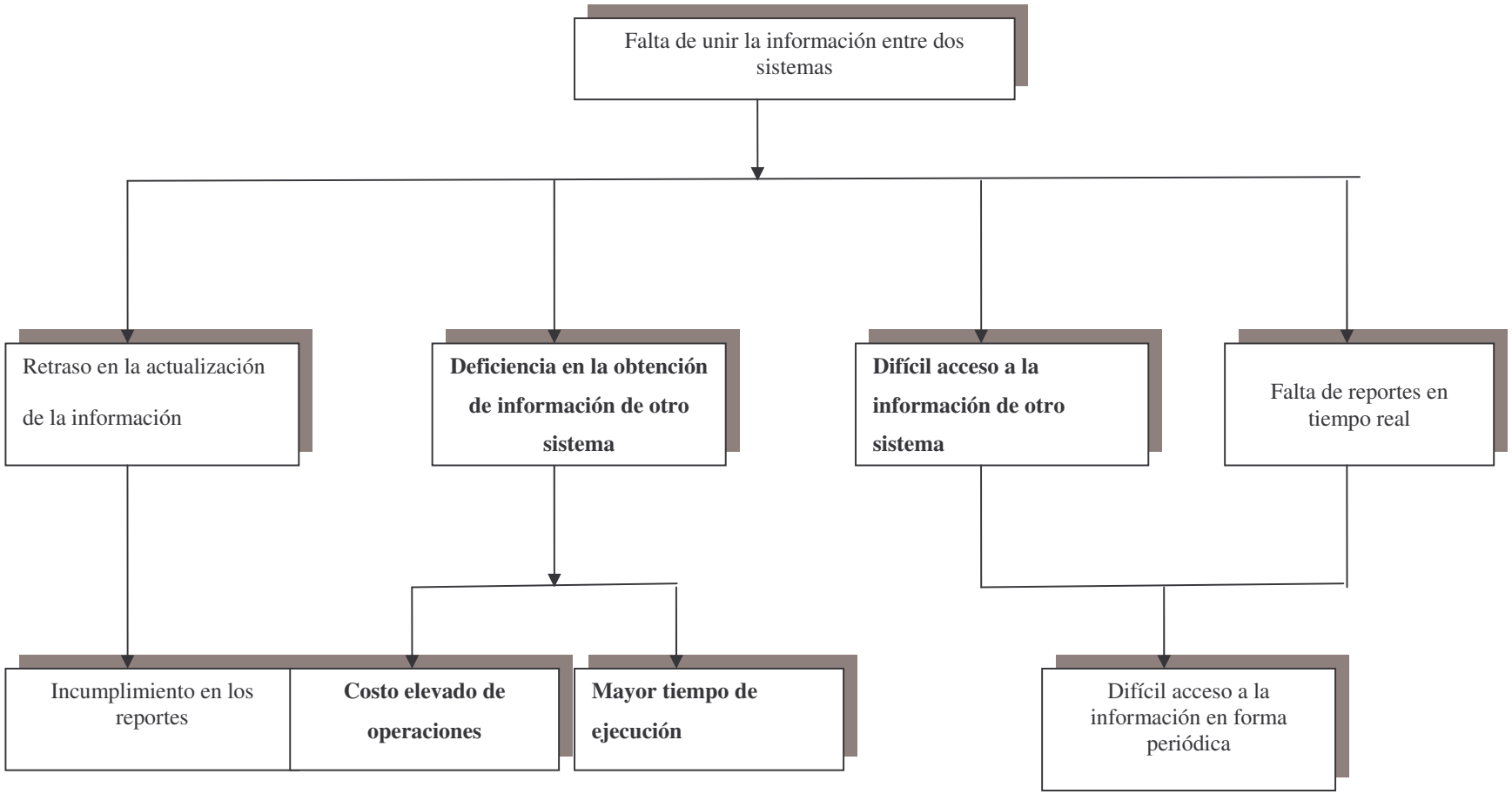
Características	Descripción
POR.1 Adaptabilidad	Capacidad del producto de software para ser adaptado a diferentes ambientes especificados sin aplicar acciones u otros medios que no sean los provistos para este propósito en el software considerado.
POR.2 Co-existencia	Capacidad del producto de software para co-existir con otro software independiente en un ambiente común compartiendo recursos comunes.
POR.3 Capacidad de remplazo	Capacidad del producto de software para ser usado en lugar de otro producto de software especificado para el mismo propósito en un mismo ambiente. Por ejemplo, la capacidad para el reemplazo de una nueva versión de un producto es importante para el usuario, cuando ésta se actualiza.

Tabla Anexo 6. Características del Modelo correspondiente a la categoría Portabilidad.

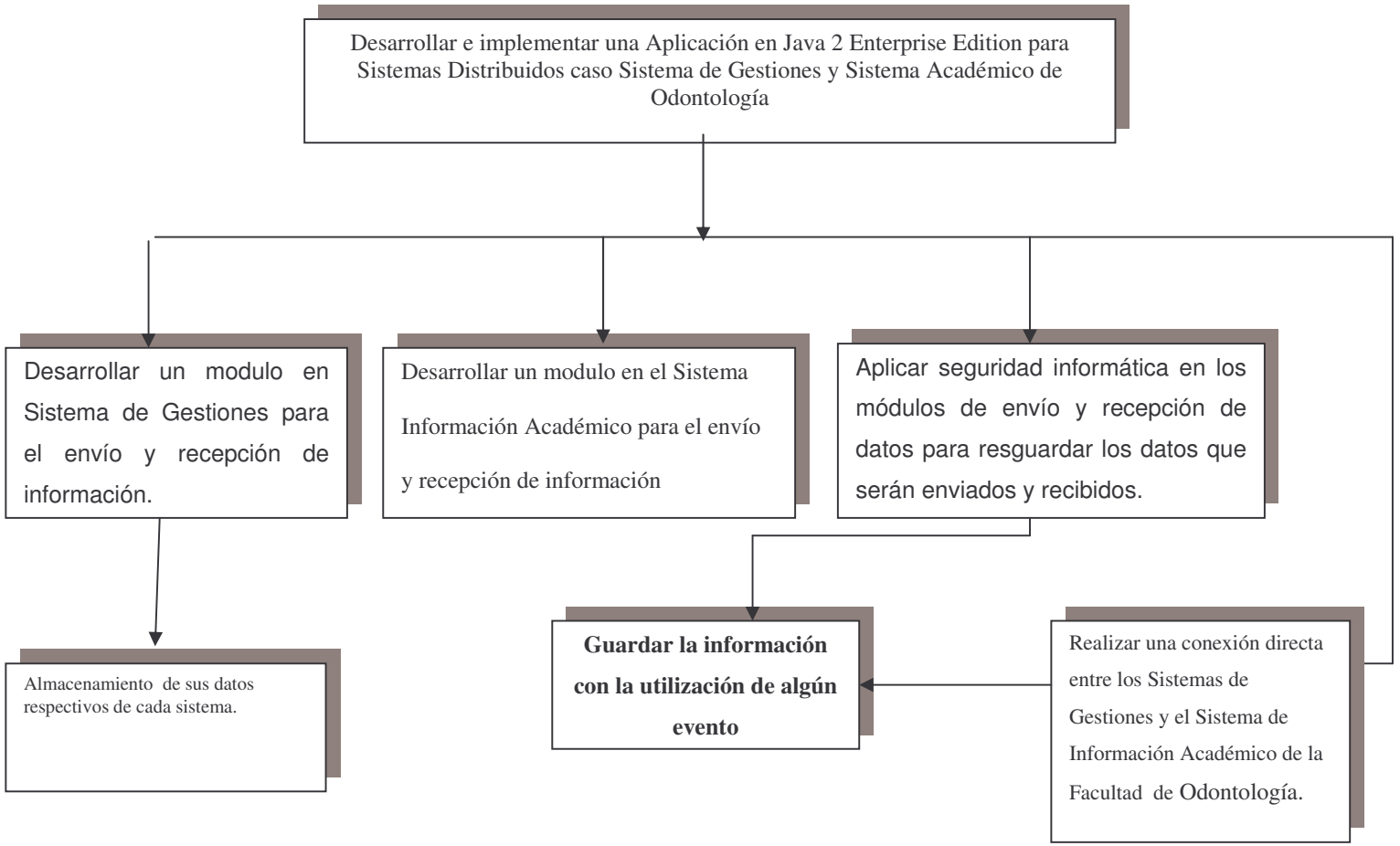
MARCO LOGICO

Resumen Narrativo.	Indicadores Objetivamente Verificables	Medios de Verificación	Supuestos
Fin del programa: Confiabilidad y seguridad en el tratamiento de la información para realizar procesos en línea en los sistemas distribuidos y su mejoramiento con la toma de decisiones.	- Mejoramiento en manejo de información. - Disposición de más recursos. - Información disponible en tiempo real para reportes.	- Base de Datos actualizados en línea. - Reportes.	
Propósito: Aplicación del Java 2 Enterprise Edition para Sistemas Distribuidos Caso: Sistemas de Gestiones y Sistema Académico de Odontología.	- El software concluido para Noviembre de 2007 - No existen los procesos manuales en la migración de datos de un sistema a otro. - Actualización de los nuevos alumnos matriculados en una determinada carrera son conocidos en tiempo real.	- Software almacenado en un directorio del servidor de aplicaciones de la UMSA. - Documentación existente sobre el Proyecto para la UMSA. - Listado de código impreso en Papel.	- Existe documentación histórica de los alumnos - Cumplimiento con los respectivos requisitos de la carrera y de gestiones de la UMSA.
Producto: 1. Una Aplicación para actualizar la información en sistemas distribuidos. 2. Modulo de Seguridad en el envío y recepción de datos para garantizar la actualización de la información.	- Prueba de Caja Negra a la aplicación desarrollada - Prueba de Caja Negra a los modulo de los sistema con lo que va intervenir l aplicación.	- Listados de estudiantes matriculados en la carrera de Odontología. - Listados de Estudiantes egresados en el sistema de Gestiones	- La institución no abandone el proyecto - La UMSA brinde un espacio físico para la implantación del sistema. - Mejor manejo y administración de la información por parte de la UMSA y la Carrera de Odontología.
Actividades: 1. Estudio del escenario de trabajo 2. Identificar la necesidades y limitaciones. 3. Definición de requisitos. 4. Desarrollar una aplicación para este propósito. 5. Definir herramientas para desarrollo del sistema. 6. Desarrollar métodos de protección para la seguridad del sistema de información desarrollado.	**Actividad — Tiempo/Costo** Equipo de Computación — 1 semana/1500 $us Material de escritorio — 1 semana/100 $us Personal para el diseño y desarrollo — 1 semana/1200 $us Estudio Preliminar — 1 semana/10 $us. Recolección de información. — 1 semanas/30 $us Determinar las tareas a desarrollar. — 1 semanas/10 $us. Análisis y diseño del sistema. — 1 semanas/10 $us Sistemas a desarrollar — 1 semanas/10 $us Demostración de prototipo — 1semana/10 $us. Desarrollo del sistema — 2 semanas/200 $us Instalación del sistema y pruebas. — 1 semanas/10 $us.. Entrega y puesta en marcha del sistema. — 1 semana/100 $ ***Tiempo Total 3 meses y 1 semana/ 3070 $us***	- Facturas de compra de material de escritorio e insumos. - Informes periódicos de avance del Sistema a la UMSA - Carta de Conformidad de las Autoridades de la UMSA como evaluador del proyecto para la UMSA.	- Los involucrados predispuestos al trabajo conjunto. - Personal de la UMSA dispuesto a fortalecer su conocimientos informáticos. - Se cuenta con el acceso a toda la información. - Se cuenta con la asistencia de otros profesionales.

ARBOL DE PROBLEMAS:



ARBOL DE OBJETIVOS:



BIBLIOGRAFÍA

[1]. [PRES, 2000] Roger S. Pressman. 2000: Ingeniería de Software, Quinta Edición,
Editorial McGraw - Hill-México.

[2]. [INST, 2004] Instituto Tecnológico de Colima 2004: Tutorial de Sistemas Distribuidos, Departamento de Sistemas y Computación.
http://www.itcolima.edu.mx/profesores/tutoriales/sistemas_distribuidos_I/index.htm

[3]. [ARAU, 2004] Sistemas Distribuidos 2004: Sistemas Distribuidos
Concepto y Características, Alfonso Araujo Cárdenas
http://mx.geocities.com/alfonsoaraujocardenas/sistemasdistribuidos.html

[4]. [BAÑA, 2006] Sistemas Distribuidos 2006: Conceptos y estándares de arquitecturas orientadas a servicios Web, Master: José Angel Bañares Bañares
UNIVERSIDAD de ZARAGOZA
http://iaaa.cps.unizar.es/docencia/SWDoct.html

[5]. [SERV, 2001] Servicios Web 2001: Que es un servicio Web?, MC Carlos Lizárraga Celaya, Universida de Sonora
http://www.fisica.uson.mx/carlos/WebServices/WSOverview.htm

[6]. [DBMS, 2005] DBMS - Sistemas Distribuidos 2005: Sistemas de Bases de Datos Distribuidos - DBMS - Técnicas de acceso, Algebra Relacional, estandar SQL2 y otros. Córdoba Argentina
http://monstera.man.poznan.pl/wiki/index.php/Mysql_vs_postgres

[7]. [ALVA, 2006] Desarrollo Agil con J2EE 2006: Desarrollo Agil con J2EE con herramientas OpenSourse, Jose María Alvarez Rodrigues, Artículo

[8]. [SCHM, 2005] Aprendiendo UML 2005: Aprendiendo UML en 24 horas, Prentice Hall, Schmuller Joseph

[9]. [VILL, 2001] Introducción al XML 2001: Introducción al XML, Universidad de Oporto, Jaime E. Villate.
http://www.oasis-open.org/cover/sgml-xml.html

[10]. [GERM, 2003] XML 2003: Daniel M. German,
http://www.solomanuales.org/curso

[11]. [G&L, 1997] Glasinovic y Landivar 1997: Redes de Comunicación y Base de Datos Distribuidas Sistemas de Bancos, Tesis.

[12]. [K&K, 2002] Kendall & Kendall 2002: Análisis y Diseño de Sistemas, 913 Tercera Edición Editorial Amy Cohen, México.

[13]. [NAV, 2002] Navarro Gutiérrez E. 2002: Tratamiento de Apoyo al Registro Administrativo obre la Plataforma Internet y Base de Datos Distribuidas, Caso Instituto Nacional de Estadística, Proyecto de Grado, La Paz – Bolivia.

[14]. [TIC, 2003] Ticona Gutierrez A. 2003: Sistema Integrado de Información Administrativo Financiero, Proyecto de Grado, La Paz – Bolivia.

[16]. [PEÑ, 2004] Peñafiel Plata G. A. 2004: Gestión Académica vía Web Universidad Franz Tamayo, Proyecto de Grado, La Paz – Bolivia.