UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INFORMÁTICA

"DESARROLLO DE PÁGINAS DINÁMICAS (JSP) EN INTERNET UTILIZANDO APACHE"

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA

Carlo Franco García Sánchez

DIRIGIDA POR:

M. C. Ma. Teresa García Ramírez.

Santiago de Querétaro, Qro., 10 de Mayo de 2005.

RESUMEN

El modelo Model View Controller MVC es fundamental en el desarrollo de aplicaciones porque permite trabajar independientemente en la implementación del Frontend que incluye el uso de animaciones con dibujos vectoriales, ActionScript para eventos, optimización de imágenes, sonido, video y código HTML que se integra en un solo elemento visual; y la implementación del Back-end que incluye la programación que se requiere para el manejo de la base de datos y el control de flujo de la información. Una de las herramientas que soporta el modelo MVC es Java que cuenta con la tecnología necesaria, como JavaBeans, servlets y JSP's, que permiten interactuar y adherirse con el Front-End para crear una aplicación completa, versátil y atractiva.

DEDICATORIA

Este trabajo, que es el fruto de años de esfuerzo y el resultado de la formación que tengo no solo en el aspecto académico sino que también como persona, refleja la necesidad que tengo de aprender y de seguir en el camino de la formación individual apoyándome como siempre en las personas que me quieren y me han dado soporte a pesar de mis fallas.

Por esto, reitero mi camino de vida dedicando éste trabajo a las personas que me han apoyado durante toda mi existencia dándome retos difíciles y satisfacciones así como una inmensa felicidad, va para ustedes mi familia

GRACIAS MAMÁ Y GRACIAS HERMANO

AGRADECIMIENTOS

Es preciso dar crédito a las personas que lo merecen ya que sin ellos no podría haberse llevado a cabo éste proyecto:

- A ***DIOS, Constructor de mi Camino,*** que me ha llevado por rutas sinuosas que se traducen en recompensas y aprendizajes.
- A ***Mi Mamá,*** que gracias a su carácter y amor he podido realizar éste trabajo.
- A ***Mi Hermano,*** que agradezco el haber podido recuperar algo del tiempo que no habíamos aprovechado.
- A ***Tere,*** ya que me abrió las puertas de la confianza y del trabajo, para darme una prueba de éxito, para que lo busque de aquí en adelante.
- Un agradecimiento especial para dos personas que han contribuido de una manera significativa para que se desarrolle éste trabajo y que me han enseñado la importancia de la comunicación, la amistad y la fortaleza ante las situaciones difíciles GRACIAS ***Adriana*** y ***Ricardo***.
- Un agradecimiento por la enseñanza de vida que me ayuda a vivir como soy y por los momentos especiales que hemos vivido, gracias ***Viel*** y ***Toño***.
- Ahora es importante reconocer a las personas que motivan la vida de otras, gracias a esas personas especiales, mis ***AMIGOS,*** que sin ellos, simplemente no tenemos motivación por destacar, gracias a todos.

ÍNDICE

Índice de Figuras

Índice de Tablas

1. INTRODUCCIÓN

En los últimos años el avance tecnológico ha permitido que las personas ya no tengan la necesidad de desplazarse para desarrollar sus actividades normales, como ir a su lugar de trabajo, de compras, hacer transacciones bancarias, etcétera; ya que estas actividades se pueden llevar a cabo por medio de una conexión a Internet y una aplicación que permita efectuar las distintas operaciones que se requieran desde cualquier sitio; un ejemplo de dichas operaciones pueden ser, en el área laboral, una reunión con los gerentes generales de una empresa que tiene varias sucursales a lo largo de un país; ir navegando por la red de un supermercado e ir seleccionando los distintos productos necesarios para el hogar, llenar formularios para la solicitud de algún servicio o bien y hasta realizar transacciones bancarias, independientemente del lugar en el que se encuentre o la plataforma sobre la que se esté trabajando.

Por lo anterior Internet es un medio importante para la transmisión de información en los rubros de diferentes áreas como la difusión tecnológica, promoción de productos y servicios, más recientemente, la transmisión de aplicaciones completas y robustas que proporcionan una gama de procesos diversos a través de la red.

Las aplicaciones distribuidas por Internet llamadas aplicaciones Web, proporcionan una versatilidad importante en un sistema, ya que pueden estar distribuidas en diferentes regiones. Una aplicación Web, consiste en un catálogo de páginas que interactúan entre sí ya sea de forma estática, es decir cuando no se modifica el contenido de las páginas, o de forma dinámica, es decir cuando a través de una base de datos se obtiene la información que será desplegada en las páginas de Internet.

Objetivo de la Tesis

El presente trabajo tiene como objetivo el análisis e implantación de páginas dinámicas en internet con JSP haciendo uso del sistema operativo Linux y el servidor web Apache. Para

lo cual se requiere realizar ciertas actividades como son la instalación y configuración del Sistema Operativo Linux Red Hat 8.0, el servidor Apache, el servidor Tomcat y el lenguaje Java.

Justificación

La tecnología Java brinda una amplia gama de posibilidades para desarrollar cualquier tipo de aplicación de acuerdo con las necesidades del cliente, tales como aplicaciones para PC o personales, en red e Internet principalmente; Java es un lenguaje versátil que puede ser utilizado para la realización de páginas en Internet, las cuales son almacenadas y presentadas por un Servidor Web, permitiendo que sean vistas por cualquier persona que acceda a la página por medio de un explorador Web.

Para el desarrollo de una aplicación Web con calidad es importante seguir un modelo de desarrollo que garantice el trabajo en equipo, la terminación del producto en tiempo y procedimientos de evaluación de la calidad durante todo el proceso. Entre los modelos propuestos para el desarrollo de aplicaciones Web esta el MVC (Model View Controller) que permite separar la lógica de la aplicación (procesos) de la interfaz o vista de la aplicación.

Así mismo, la implantación de este tipo de aplicaciones, requiere de una variedad de herramientas, que van desde la plataforma, el sistema operativo, el contenedor de la aplicación (servidor web), el almacenamiento de la información y el software necesario para cada etapa de desarrollo. En este trabajo, se utilizó una PC con sistema operativo Linux, el servidor web Apache y la base de datos MySQL, cuyas características se describen en el presente documento. Para la etapa de desarrollo de la vista se usó software para el diseño como Dreamweaver en la codificación de las páginas Web, Flash para los elementos multimedia, Fireworks para la creación de dibujos y Freehand para el tratamiento de imágenes. Finalmente para el desarrollo de la lógica de aplicación (procesos) se utilizó la tecnología Java y Jakarta en sus derivados JSP y Tomcat respectivamente, los cuales se abordan a lo largo de este escrito. Además se utiliza el MVC para llevar un

control del diseño y desarrollo.

En esta investigación se analizará la tecnología java para el desarrollo de páginas dinámicas, sobre el Servidor Web Apache y la plataforma Linux, y con esto observar el grado de seguridad que se logra.

Contenido de la tesis

En cuanto a la estructura de este trabajo, se contemplan siete temas divididos en capítulos, los primeros 3 capítulos abordan lo respectivo a introducción, sistema operativo Linux y servidor web Apache, en los capítulos del 4 al 6 se abordan las herramientas para el desarrollo de la aplicación como son el lenguaje de programación Java, el servidor Jakarta Tomcat para páginas JSP y la base de datos MySQL; en el capítulo 7 se describen cada una de las etapas para la implantación de la aplicación, siguiendo como patrón de análisis y diseño el MVC. Finalmente en el capítulo 8 se exponen las conclusiones que se derivan de la investigación y el desarrollo de la aplicación planteada, así como los posibles trabajos futuros.

2. SISTEMA OPERATIVO LINUX

Los Sistemas Operativos que existen en la actualidad, son muchos y todos muy diferentes, están por ejemplo los de Microsoft que abarcan una parte muy extensa en el Mercado, teniendo diversos Sistemas Operativos como Windows 98, Windows ME, Windows 2000 y Windows XP; y por otro lado existe LINUX, con otra gama de distribuciones tales como Linux Mandrake, SUSE y Red Hat, pero entre toda esa gama de distribuciones surge la necesidad de establecer un ambiente en donde los usuarios no deben de ser afectados por las mismas que en el mercado existen, para poder ejecutar cualquier aplicación que sea necesaria sin importar la plataforma que el usuario prefiera.

Este trabajo, se desarrolla con base al Sistema Operativo Linux Red Hat 8.0, ya que es necesario trabajar con Servidores de Archivos, Servidores Web o Servidores de FTP e incluso Servidores de Noticias y de Correo electrónico, los cuales se encuentran en esta distribución, para el desarrollo de sistemas de información distribuidas.

2.1 Características

Para comenzar a hablar de Linux Red Hat es necesario establecer algunas de sus características, que se aplican para otros sistemas operativos de la misma familia Linux, ya que este Sistema Operativo se deriva de la combinación de UNIX y WINDOWS, el primero se utiliza para el desarrollo tecnológico y tiene varias características, entre ellas el tiempo compartido y el procesamiento en paralelo, UNIX es muy costoso y por el otro lado Windows que es un sistema operativo comercial que se plantea para la interfaz e imagen con el usuario teniendo características que lo hacen amigable para la configuración del mismo, pero crea una dependencia de usuario. Teniendo como referencia a estos dos sistemas, UNIX, Windows, nace Linux que combina estas dos cualidades y se crea e implementa tratando de dar a conocer la cultura del software libre.

El creador de Linux, Linus Torldvards, quien da la pauta para esta gran cultura del software libre, desarrolla un kernel completamente en C y lo comparte en una comunidad de hackers

e investigadores para que éstos a su vez puedan mejorar e implementar este kernel hasta llegar al potente software de distribución libre que hoy conocemos como Linux, en sus diferentes versiones y para las distintas aplicaciones para las que se especializa.

Linux cuenta con características que lo hace en particular un Sistema Operativo confiable y seguro para el usuario final. Éstas se describen a continuación.

A. ***MULTIUSUARIO,*** esto permite que la misma computadora, tenga varios usuarios, y dependiendo de las características de la misma, se pueden estar atendiendo a estos usuarios y despachando recursos al mismo tiempo sin tener que estar dependiendo de la máquina para trabajar, sino que desde cualquier estación, se puede conectar un usuario a su sesión en una máquina con Linux y poder trabajar estando en su sesión sin depender de tener que trabajar en una computadora personalizada y ubicada en un lugar específico, lo cual da independencia de recursos por medio de terminales tontas esto se puede observar en la Figura 1.

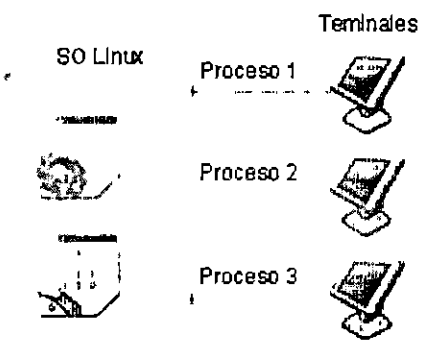

Figura 1. Proceso Multiusuario.

Las terminales tontas, son aquellas que dependen totalmente de un servidor que les proporcione el sistema de archivos, acceso a aplicaciones y a comunicaciones, sin contar éstas con un disco local.

B. ***MULTIPROCESO,*** Linux puede realizar varios procesos o aplicaciones a la vez, puede estar ejecutando un servidor DNS, HTTP y SMTP a la vez, atendiendo peticiones de

éstos al mismo tiempo.

C. ***MULTIPLATAFORMA***, Linux es compatible y funcional con la mayoría de las plataformas en el mercado algunas de ellas son 386, 486, Pentium, Motorola680, Sun Sparc, entre otros [Negus, 10].

D. ***SHELLS*** programables, que permiten configurar y programar el sistema operativo de tal forma que el sistema será totalmente amigable para el usuario. La interacción con el usuario del sistema operativo se muestra en la Figura 2.

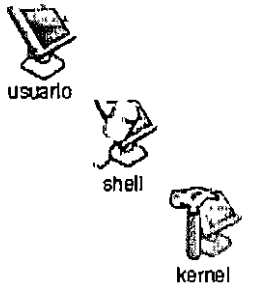

Figura 2. Estructura Linux.

E. ***SERVIDORES DE RED***, esto se refiere a la especialidad de Linux, que es ofrecer un ambiente totalmente orientado a las comunicaciones, con los que se pueden realizar aplicaciones en ambientes Distribuidos o ambientes de Interconexión de Redes.

2.2 Ventajas

Las ventajas que proporciona un Sistema Operativo como Linux, son diversas, las cuales se describen a continuación.

A. ***Confiabilidad***, Linux es muy íntegro, ya que no se requiere reiniciar el sistema cuando se realizan mejoras y actualizaciones en el mismo; esto es una gran ventaja, ya que cuando el Sistema está siendo un servidor de archivos o un servidor Web y no tiene que detenerse para que se realicen las actualizaciones o cambios al sistema operativo.

B. ***Tutoriales***, que se refiere a que como el sistema operativo es de libre distribución, mucha gente lo tenga, y así, cualquier problema o duda que surja en torno al software, siempre hay a quien recurrir.

C. ***Open source***, La libre distribución de un Sistema Operativo confiable y totalmente claro para su desarrollo e implementación, le da una ventaja enorme, ya que no es necesario gastar enormes cantidades de dinero para obtener cualquiera de las distribuciones de Linux, ya que aún cuando el software se vende, no cuesta más de 1000 pesos.

2.3 Administración

Para llevar a cabo una administración en un sistema Operativo Linux, es necesario tomar en cuenta que varias personas pueden tener acceso al sistema al mismo tiempo, y que no todas tendrán el permiso para modificar los valores de administración y configuración del sistema, como lo es la administración de carpetas, la configuración de la seguridad, la administración de los distintos usuarios y los permisos de los mismos. Por esto hay un usuario del Sistema que es el usuario root o el administrador del sistema y es quien determina los valores de configuración para el Sistema Operativo.

Para cambiar a modo de administrador, en una Terminal, se escribe el comando *su*, y se introduce la contraseña de administrador para convertirse en el usuario root. El comando *su*, significa switch user, puede cambiar el shell o los valores definidos de una cuenta de usuario en sesión a cualquier usuario que esté dado de alta en el sistema haciendo referencia a un comando con el nombre del usuario al que se pretende acceder, su sintaxis es *su – nombre de usuario*; en caso de no especificar un usuario en particular, se determina que se quiere cambiar al modo de root.

El modo de root es el que va a permitir llevar a cabo la administración del sistema.

Para que diferentes usuarios tengan acceso a las diferentes aplicaciones instaladas por el usuario root, es necesario que cada uno tenga dadas de alta las variables de entorno que

apunten a los directorios de éstas. Para que todos los usuarios tengan acceso a los diferentes programas, se configura el archivo *profile*, el cual determina los valores de las variables de entorno para todo el sistema operativo, en cambio si se desea personalizar las aplicaciones para un usuario en específico, se modifica el archivo *bash* que tiene en su directorio.

En éste proyecto, se instala la máquina virtual de JAVA y el Tomcat y se dan de alta las variables de entorno en el *profile* para que todos los usuarios dados de alta puedan utilizar las herramientas.

2.3.1 Comandos administrativos

Los comandos a los que los usuarios están ligados y limitados, son los comandos a los que son asignados en su path. En el caso de root, se le pueden asociar comandos que son para la ejecución de tareas importantes como lo son fdisk, grub y el comando init, que se encuentran en el directorio /sbin.

Para el particionamiento del disco duro durante la instalación, exísten en modo gráfico dos ambientes de configuración el fdisk y el grub, los cuales dan la posibilidad de particionar el disco duro para la instalación del sistema operativo, se recomienda utilizar disk fruid, como interfaz para la configuración, en donde se proporcionan las diferentes particiones para intercambio, mapeo de memoria y la raíz del sistema operativo.

En el directorio /usr/sbin se localizan los comandos para administrar las cuentas de usuarios, la configuración del mouse y del teclado y muchos de los procesos que se ejecutan como demonios.

Algunos comandos se guardan en archivos /bin y /usr/bin para que estén disponibles para los usuarios normales, y también para root, por ejemplo, el comando mount, es accesible para los usuarios y pueden montar cualquier unidad, pero solo root puede montar un sistema de archivos, que utiliza el mismo comando.

En el presente trabajo se crean dos usuarios de la computadora, cada uno con privilegios de

administración por medio de intercambiar de shell a root por medio del comando *useradd.*

2.3.2 Archivos de configuración

Los archivos de configuración tienen una gran ventaja, que consiste en que son archivos de texto sin formato, lo que significa que se pueden editar por cualquier editor sin dificultades de alguna incompatibilidad por parte de un sistema operativo o paquete de edición de texto.

Los archivos principales de configuración de servicios se encuentran en el directorio de /etc, dependiendo del servicio que se quiere configurar, por ejemplo, se tiene el control de la configuración del servidor Web, ya que en este directorio se encuentra la configuración de todo el servidor Apache, los directorios de configuración, módulos y librerías que utiliza. Los archivos de configuración del servidor de FTP, que describe los parámetros de funcionamiento de este servicio. La configuración del sistema X, que es el sistema que controla la parte de los gráficos y la resolución para que se tenga un óptimo funcionamiento de la interfaz gráfica.

Un punto que no se debe olvidar es la configuración de la seguridad y la privacidad para con los usuarios del sistema, que en este directorio de /etc, es en donde se almacenan los archivos que guardan la configuración e información de cada cuenta que se da de alta en el sistema, estos archivos, trabajan por decirlo de algún modo de forma conjunta, uno se llama passwd y el segundo se llama shadow.

Otro archivo de configuración es el que concierne al demonio de impresión, que es el archivo LPD, que nos determina la configuración de la impresora que está determinada o para configurarla o modificar la configuración actual.

Otro archivo muy importante que no se debe de obviar es el archivo que controla todo el entorno de variables y de rutas para el funcionamiento completo del sistema operativo y determina las aplicaciones que estarán disponibles para los usuarios, así como la ruta de los diferentes directorios donde se encuentran los ejecutables de las aplicaciones instaladas, para poderlos utilizar sin marcar un path, antes del comando y cambiar de directorio cada

vez que se requiera algún comando específico. Este archivo es el PROFILE, que guarda la configuración de los shells públicos o los que estarán disponibles para cualquier usuario que ingrese al sistema además, se configura con las variables de entorno del sistema para las aplicaciones a utilizar, teniendo en cuenta los privilegios de los directorios y las aplicaciones.

2.4 Seguridad

La seguridad en cualquier sistema operativo es vital para el ciclo de vida y funcionamiento de una aplicación. En linux, la seguridad se aborda de una manera delicada y muy libre para el usuario o administrador del sistema, ya que éste determina hasta donde quiere estar protegido en su sistema, sin embargo, si el administrador no tiene la suficiente experiencia y conocimiento, puede dejar muy vulnerable el sistema de tal forma que alguien que desee irrumpir en el sistema para corromperlo al realizar actos de piratería o crackeo.

La seguridad que se maneja en el sistema operativo Linux Red Hat, se define desde el momento de instalación al configurar un nivel de seguridad, que establece el grado de seguridad se va a tener en el sistema, lo cual permite configurar los servicios que estarán autorizados como los servicios del protocolo TCP/IP: FTP, HTTP, SMTP, etc.

En cuanto al manejo de las contraseñas para los diferentes usuarios registrados en el sistema, se utilizan dos tipos diferentes de archivos, uno que puede ser accedido por los diferentes usuarios instalados en el sistema; y otro que solo lo puede acceder el usuario root.

2.4.1 Passwd

El primer archivo passwd, contiene toda la información referente a los diferentes usuarios y recursos del sistema, éste archivo, contiene la ruta en donde se localizan los diferentes archivos, los tipos de shell sobre los que trabajan, número de identificación que asigna el usuario o el sistema, el número de grupo; alguna descripción o nombre, y el campo que concierne a la contraseña, no se muestra en este archivo.

2.4.2 Shadow

El archivo shadow, es la protección del archivo passwd, ya que contiene el resultado de un cifrado del password de cada uno de los usuarios. Ésta sombra del passwd, sólo es accesible para el administrador, dándole una mayor seguridad al sistema, ya que cualquiera que este buscando alguna contraseña, no será accesible por medio del passwd.

En segundo punto, la contraseña que se almacena al dar de alta a un usuario del sistema, no es la que éste teclea, sino que se almacena un resultado de una operación para cifrar la palabra que se escriba. Así cada vez que el usuario quiere ingresar al sistema, no lo hace directamente colocando la palabra que es su contraseña, sino que se realiza la operación de cifrado, y la palabra resultante es la que se compara con la palabra almacenada.

Para aumentar la seguridad del sistema operativo, se activa la protección del *passwd* por medio del *shadow* con el comando *passwd conv*; con lo cual aseguramos la protección del sistema operativo.

Otro de los aspectos modificados dentro del sistema operativo para dar seguridad, es dar autorización del uso de las aplicaciones en las diferentes capas de red que proporciona el sistema. Para el proceso de Apache, se da de alta con el comando *service* y el servicio de FTP por medio del comando *chkconfig*.

3. SERVIDOR WEB APACHE

El servidor Web apache, se ha convertido en uno de los más confiables y seguros para el diseño de proyectos enfocados al desarrollo de páginas Web en las diferentes plataformas. Este servidor es de distribución gratuita, fue creado por la participación de personas que han contribuido a mejorar y desarrollar aplicaciones en Internet. En las distribuciones Linux está incluido el Servidor Web Apache.

El Servidor esta creado en código C, se compila con directivas y módulos que vienen en el paquete. En las versiones más recientes de Linux existen módulos mínimos para el funcionamiento del servidor, tal es el caso de la distribución de Linux Red Hat 8, donde los módulos core.c, prefork.c, http_core.c y mod_so.c, están compilados. Se pueden agregar más módulos para que el servidor pueda realizar funciones más especializadas y diversas, como lo son la autentificación, los proxys, entre otros.

3.1 Características

Entre las características principales del servidor Web apache están: la estabilidad ya que tiene un manejo de mensajes para los posibles errores que pudieran surgir, sin que se detenga el servidor por cualquier causa, el rápido mejoramiento del mismo, gracias a la participación de muchos programadores que están en el proyecto, es gratuito y se deja abierto a descargar completo el código fuente y así tener la oportunidad de modificarlo y ajustarlo a los requerimientos de la aplicación, es muy fácil de configurar ya que sus archivos de configuración se basan en documentos de texto sin formato y son modificables en cualquier editor; puede llevar a cabo un control de acceso de diferentes maneras, ya sea un bloqueo por las direcciones IP, por nombre de host o por el método de usuario y contraseña. Otra característica importante y destacada de apache, es que tiene un API personalizado que permite la inclusión de módulos externos para que el servidor los utilice[Negus, 10].

3.2 Ventajas

Las ventajas de tener un Servidor Web Apache, son principalmente que es un software libre y de distribución gratuita, lo cual da un amplio margen de desarrollo, ya que se puede obtener el código fuente y así, poder modificar o utilizar el Servidor como mejor convenga a las necesidades particulares de cada institución o aplicación; otra de las ventajas de Apache, es que se pueden construir módulos específicos que requieren de cierto comportamiento en una situación específica para el servidor, así, apache viene con una herramienta llamada *apxs*, que es la que permite compilar un módulo para agregarlo a la configuración del servidor, para responder a los requerimientos importantes en la aplicación; La seguridad es otra ventaja que maneja el servidor cuando está en un Sistema Operativo como Linux, que consiste, principalmente, en que el directorio de configuración, que se encuentra en el directorio /etc, solo esta a la vista del Administrador del sistema y del administrador del Servidor Web, así, ningún usuario mas podrá cambiar o modificar la configuración del Servidor; una ventaja más del Servidor Apache es que el demonio, ya puede realizar trabajos multitarea, ya que el proceso principal, puede tener varios procesos padres que pueden atender llamadas del puerto y a su vez, éstos pueden tener varios procesos hijos para resolver las necesidades de las peticiones, así, varios procesos pueden estar realizando varias tareas a la vez.

En si el servidor es fácil de configurar, expandir, administrar, tiene varios modos de seguridad, puede ser multitarea, multihilo o multiproceso.

3.3 Administración

La administración del servidor Web Apache, se realiza mediante la configuración a los archivos del directorio HTTPD, que se encuentra dentro del directorio de configuración del sistema /etc.

Dentro de /etc, se puede encontrar el directorio de apache httpd, que es en donde se encuentran los archivos de configuración y administración del servidor. Dentro de esta carpeta, existen diversos directorios de configuración y varias ligas a los directorios base, en donde están módulos, librerías, comandos de ejecución, entre otros, en la Tabla 3.0 se

describe el contenido del directorio httpd:

Tabla 3.0 Contenido del directorio httpd			
NOMBRE	*DESCRIPCIÓN*	*TIPO*	*RUTA*
Build	Contiene archivos de configuración que indica con que módulos se compiló o se compilará Apache, si no está instalado.	Liga	/usr/lib/httpd/build
Conf	Contiene los archivos de configuración del servidor, o las funciones que realizará, como las realizará y cómo se comportará.	Archivo	/etc/httpd/conf
conf.d	Contiene archivos de configuración especiales de apache, como lo son la configuración para php, bases de datos y ssl.	Archivo	/etc/httpd/conf.d
logs	Contiene los archivos de registro del Servidor, ya sean de errores, accesos y peticiones.	Liga	/var/log/httpd
modules	Contiene los módulos que se han agregado al Servidor para que realice funciones específicas.	Liga	/usr/lib/httpd/modules
Run	Contiene archivos y directorios que especifican el número de procesos que esta utilizando el servidor para su funcionamiento.	Liga	/var/run

En el directorio /etc/httpd, es en donde se realiza la configuración del servidor, para adecuarlo a las necesidades de la aplicación o institución.

Dentro del directorio conf de apache, se tiene el archivo httpd.conf, que es el archivo principal de configuración de apache, éste se divide en tres secciones, el entorno global del servidor, la de configuración del servidor principal y la definición de hosts virtuales, que describiremos a continuación:

En la configuración del entorno global se establecen las directivas necesarias que van a afectar a la actividad general del servidor. La configuración de las directivas está agrupada en tres secciones:

1.- Las directivas que controlan los procesos de operación del Servidor Apache como un

todo ("El ambiente global").

2.- Las directivas que definen los parámetros del servidor, ya sea del servidor principal o del preestablecido, para que responda a las peticiones que no son manejadas por un host virtual. Estas directivas, contienen también valores por default para los hosts virtuales.

3.- Valores para hosts virtuales, permiten que las peticiones Web sean enviadas a diferentes direcciones IP o nombres de hosts y puedan ser manejadas por el mismo proceso del servidor Web[Linux, 14].

3.3.1 Directivas de configuración global del servidor apache

SERVER TOKENS

Esta directiva impide que otros equipos remotos vean que componentes son los que se ejecutan en el servidor.

SERVER ROOT

Se encarga de la ubicación del directorio principal de apache, que es en donde se encuentran los directorios de configuración, de registro y de error.

SCOREBOARD

Crea un archivo que tenga un registro de los procesos internos del servidor, ésta aparece comentada en el archivo de configuración, ya que las nuevas versiones de apache guardan este registro en un segmento de memoria.

PIDFILE

Contiene la ruta del archivo de donde el servidor recupera el número de identificación de proceso principal cuando éste inicia.

TIMEOUT

Indica al servidor el tiempo que debe de esperar entre la recepción y envío de paquetes.

KEEPALIVE

Indica si se debe o no quedar abierta una conexión con el cliente que ha hecho una petición y así, si hace otra petición, no necesita realizar otra conexión para el mismo cliente ya que está abierta la conexión anterior.

MAXKEEPALIVEREQUESTS

Indica cuantas solicitudes se pueden realizar mientras una conexión está establecida.

KEEPALIVETIMEOUT

Indica cuanto tiempo se debe esperar antes de que un cliente haga una petición del cliente por la misma conexión.

3.3.1.1 Directivas de configuración del servidor apache en modo mpm

La siguiente sección del archivo de configuración, permite establecer las directivas y parámetros del servidor para que funcione de modo MPM, que es el Módulo de Multiprocesamiento, la cual permite que el servidor trabaje de un modo multitarea y multiproceso.

Existen varios modos de procesamiento MPM, que son el prefork, el worker y el perchild, que cada uno ocupa directivas similares, pero cambia en el modo de procesamiento. El modo perchild, que no funciona en todas las plataformas[Linux, 14], implementa un módulo multiproceso y multihilo; el modo de prefork, trabaja de un modo no-multihilo, solo trabaja multiproceso y el modo worker, implementa un híbrido entre el modo multiprocesos y multihilos.

Para que el servidor cambie dinámicamente el número de proceso sobre los cuales está trabajando, es necesario que existan ciertos parámetros, tales como las directivas siguientes:

StartServers

Éste parámetro indica cuantos procesos deben iniciarse; *MinSpareServers*, que indica cuantos procesos mínimo deben de estar libres, *MaxSpareServers* indica el número máximo

de procesos que deben quedar libres, *MaxCients*, especifica el número de procesos que se pueden levantar y *MaxRequestsPerChild*, es el número máximo de solicitudes que un proceso puede servir.

Listen

Permite especificar las direcciones IP, que se requiera que el servidor escuche, y así, se indican cuantas directivas listen y direcciones IP que se requieren servir, o poner el puerto que se va a escuchar, y así no tener ningún filtro de direcciones. La directiva *Include*, carga los archivos de configuración de la carpeta /etc/httpd/conf.d.

Los módulos en apache, son funciones que agregan cierta especialización o funcionalidad a ciertos aspectos o servicios, apache se compila con módulos ya establecidos y se pueden agregar otros módulos de forma dinámica, a través de la directiva *LoadModule*, a continuación se describen algunos de los módulos y su descripción; en la inteligencia de que si algún módulo no se requiere, se puede desmontar comentándolo.

mod_access

Provee control en el acceso, ya sea por IP o nombre de host o alguna otra característica según la petición del cliente.

mod_auth

Para autentificación de usuarios utilizando archivos de texto.

mod_auth_anon

Permite a los usuarios anónimos entrar a áreas autentificándose.

mod_auth_dbm

Permite la autentificación de usuarios por medio de archivos DBM.

mod_auth_digest

Para autentificación de usuarios utilizando MD5.

mod_include

Se utiliza para SSI (Server Side Include).

mod_log_config

Para el formato de los archivos de registro del servidor.

mod_env

Modifica las variables de entorno a CGI y SSI.

mod_mime_magic

Determina el tipo MIME de un archivo leyendo unos bytes del contenido del mismo.

mod_cern_meta

Es el módulo para emular cabeceras META CERN HTTP en la semántica de los archivos.

mod_expires

Da el tiempo límite para los documentos en cache utilizando la cabecera Expires HTTP.

mod_headers

Personalización de las cabeceras HTTP en las solicitudes y las respuestas.

mod_usertrack

Coloca cookies para el rastreo de la actividad del usuario en un sitio Web.

mod_unique_id

Proporciona una variable de ambiente con un identificador único para cada petición.

mod_setenvif

Permite establecer variables de entorno basadas en las características de la petición.

mod_mime

Según el nombre del archivo, si es tipo MIME, determina como va a ser procesado,

el lenguaje, la codificación, etc.

mod_dav

Este módulo le da a apache una funcionalidad de clase 1 y clase 2 de WebDAV, esta extensión del protocolo nos permite hacer modificaciones de recursos y colecciones en servidores remotos.

mod_status

Provee información de la actividad y el desempeño del servidor.

mod_autoindex

Genera índices de directorios automáticamente.

mod_asis

Envía los archivos sin colocarle cabeceras, ya que éstos tienen las propias.

mod_info

Proporciona información de la configuración del servidor.

mod_cgi

Ejecución de Scripts CGI.

mod_dav_fs

Este trabaja junto con el módulo *web_dav* y actúa como soporte y le proporciona acceso a recursos montados en un servidor de archivos de sistema.

mod_vhost_alias

Compatibilidad para la asignación de grandes hosts virtuales configurados dinámicamente[Negus, 10].

mod_negotiation

Es el módulo, que con base a los archivos de configuración type-map y multiviews, define y retorna el archivo que mejor encaja con el cliente.

mod_dir

Es el módulo que permite mostrar índices de carpetas, con base a la directiva DirectoryIndex.

mod_imap

Este módulo procesa archivos .map, remplazando la funcionalidad del programa de CGI imagemap. Así, cualquier tipo de documento o directorio configurado para usar el manejador imap-file se procesa correctamente.

mod_actions

Este módulo permite la ejecución de scripts CGI, dependiendo del método de la petición o el tipo de medio.

mod_speling

Permite la corrección de las URL's que el usuario ha podido ingresar mal, o permite un determinado error.

mod_userdir

Permite a los usuarios entrar en directorios específicos por medio de la sintaxis http://examples.com/~user/directory/.

mod_alias

Este módulo permite mapear diferentes partes del host en el árbol de archivos y para la redirección de URL's.

mod_rewrite

Este módulo, permite con base a una regla-base, rescribir una dirección (URL), que ya ha sido escrita en el pasado, al instante, sin necesidad de terminar de escribirla.

mod_proxy

Este módulo permite configurar un servidor proxy/puerta de entrada, FTP y http, en sus diferentes encapsulaciones, *mod_proxi_ftp* y *mod_proxi_http*.

mod_webapp

Este módulo permite que apache redireccione las páginas JSP a tomcat.

En la configuración del servidor que se maneja, se determinó que los valores preestablecidos para la primera sección son valores aceptables y utilizables para la aplicación que se va a realizar.

Se agrega la carga de un módulo el módulo webapp, que se explicará a detalle en secciones posteriores, para mostrar más a detalle cómo hacerlo se puede consultar el anexo B4.

3.3.2 Configuración principal del servidor

En esta sección están las directivas que permiten dar una solución a peticiones que no serán manejadas por una definición de un host virtual, aunque también contiene valores por defecto para determinados servidores configurados, los cuales se definirán más adelante.

Las directivas que en este punto, se refieren a la configuración de diferentes identificadores o nombres a través de los cuales el servidor va a atender peticiones, ya sea por medio de uno o más hosts virtuales definidos; así como la configuración de quién es el administrador del servidor, qué directorios serán accedidos, por ejemplo si un usuario va a tener acceso a su HOME desde el servidor Web o utilizar el servidor solo una o varios usuarios identificados por medio de un nombre y grupo, así se determina desde dónde el servidor va entregar las páginas que se van a mostrar, se configura la ubicación del archivo que contiene la información de los usuarios así como su contraseña, para poder acceder a ciertas secciones del servidor, se configura como va a tratar los diferentes archivos que no pueda identificar por su extensión o por su contenido.

Los primeros valores en esta sección del servidor, son las de *USER* y *GROUP*; que determinan si algún usuario o un grupo, pueden llevar a cabo el levantamiento del servidor, así como la utilización del mismo.

La directiva *ServerAdmin,* la cual indica una dirección de correo electrónico, que se

mostrará para que el usuario se pueda comunicar con el administrador del servidor en caso de que exista algún error en la ejecución, la sintaxis es *ServerAdmin dirección@dominio.com*.

La directiva *ServerName*, establece un nombre dominio para el servidor, el cual debe estar registrado; en caso de que no esté registrado, se especifica el nombre por medio de la dirección IP y así será identificado el servidor vía IP. Otro factor importante de esta directiva es la especificación del puerto por el cual el servidor va a estar trabajando.

UseCanonicalName, esta directiva determina si apache utilizará el nombre canónico o no, si esta opción está en verdadero, se utilizará el nombre que se especificó para el servidor con la directiva anterior, si no, utilizará el nombre que está dado por default al servidor en la configuración principal.

La directiva *DocumentRoot*, da el directorio desde el cual nuestro servidor sirve las páginas, o mejor dicho, es el directorio en el cual se almacenan las páginas que el servidor va a mostrar.

La directiva *DirectoryIndex*, determina los tipos de archivos que apache va a negociar y atender; por medio de la extensión de los mismos.

La directiva *AccessFileName*, establece el nombre del archivo para encontrar información de control de acceso, esto quiere decir que los directorios que aparezcan en el archivo .ht, no serán mostrados al público que entre al sitio.

Para proteger los archivos para que no puedan ser vistos por los visitantes del servidor se utilizan los siguientes parámetros.

La directiva *DefaultType*; indica como hay que tratar a los archivos que no se puedan identificar por el tipo o la extensión de los mismos. Para lo cual se especifica una serie de parámetros que definen el módulo mime_magic, el cual determina que los archivos que no

entienda, explore si son de tipo MIME y los resuelva en tal caso.

La directiva *HostnameLookups* tiene sólo un parámetro, encendido o apagado, lo cual indica si ha de almacenar el nombre o la dirección IP de los hosts que pasen por el servidor para que pueda funcionar como un DNS, lo cual no es recomendable ya que si hay muchas peticiones éste puede saturarse.

Por tanto, esta directiva está desactivada y para llevar el control de los hosts visitantes se hace por medio de archivos de registro, los cuales determinan lo que se va a almacenar.

ErrorLog, determina en donde se encuentra el directorio de mensajes de error, para que cuando algún error se presente, se almacene dentro de un archivo de error en este directorio.

La directiva *LogLevel*, controla el número de mensajes que se almacenan en el archivo de errores dentro del directorio (ErrorLog).

Para definir algunos formatos alias para usarse con directivas personalizadas, ya sea por referencia o por agente; la directiva para pasar estos parámetros es *LogFormat*.

La directiva *CustomLog*, es la que permite personalizar cómo se tiene que almacenar un mensaje de error, o mejor dicho, define el formato de cómo se almacenarán los mensajes.

Los tipos de formato en relación al registro de los errores, pueden ser combinados de forma tal que se guarden en el archivo que produjo el error, la fecha y el host, entre otros.

La directiva *ServerSignature*, coloca, en un mensaje de error o una página generada, una firma por el servidor, que es el nombre del servidor y agrega un *mailto* al administrador del sitio, por cualquier falla o algún mal procesamiento de información.

La directiva *Alias*, mapea diferentes partes del sistema de archivos que no son parte del

DocumentRoot; aclarando que se pueden agregar cuantos alias se necesiten, pero hay que definir cada uno con sus argumentos dentro de un directorio. Esto quiere decir que se tendrán otros directorios y se podrán invocar por medio de un Alias.

La directiva de *ScriptAlias*, es parecida a la directiva Alias, la diferencia, que en los directorios de *ScriptAlias*, hay código ejecutable y por lo regular, se usa para CGI's.

Para el tratamiento de índices y de íconos, apache debe considerar como va a tratar la información, ya que éste define cuanto espacio ocupa el archivo, los íconos que va a mostrar, qué formato van a tener, si la página tendrá información acerca de la misma y en cómo se van a indexar los archivos que estén en el servidor, para hacer esta labor se usa la directiva *IndexOptions*, que funciona en conjunto con la directiva AddIconByEncoding, y la directiva AddIconByType y AddIcon.

Directivas que especifican ubicación de íconos y descripción de cómo tratar los archivos, es decir, si hay un archivo zip que esté en el sitio, el índice lo mostrará como un icono, su nombre, tamaño y además su tipo o formato.

Las directivas *HeaderName* y *ReadmeName*, son directivas que si se encuentran dentro de un directorio, los archivos se colocarán al inicio de un índice dentro de un directorio, ya que son archivos que un usuario debe leer antes de continuar.

La directiva *AddEncoding*, determina los formatos de compresión que se pueden manejar a través de un navegador al momento de recibirlos.

La directiva de *AddLanguage*, permite establecer el tipo de idioma de determinado archivo, dependiendo del sufijo del mismo; esta directiva interactúa con la directiva *LanguagePriority*, que da una resolución, en caso de que un archivo exista en varios idiomas y que el usuario no especifique tal, esta directiva toma el valor establecido y manda el archivo en el idioma para el que tiene prioridad.

Es necesario, definir con base a un estándar, el conjunto de caracteres aceptados para poder tratar diferentes tipos de páginas y de información, el servidor Web, permite configurar varios estándares para el funcionamiento de éste, uno es el estándar principal, que se define con la directiva *AddDefaultCharset*, y se agregan más estándares por medio de la directiva *AddCharset*, que define estándares de tipo ISO.

En el servidor se puede definir qué es lo que se va a realizar con archivos de determinada extensión, lo que permite identificar el archivo; esto se realiza por medio de la directiva *AddType*, que indica el tipo de archivo que se va a tratar por medio de la directiva *AddHandler* y que con base a la extensión del archivo se realizan determinadas acciones.

En apache, se pueden personalizar las respuestas de error que se generan, ya sean en texto plano, que sean redirigidas a otra página en el servidor local o externo, esto se realiza por medio de la directiva *ErrorDocument*.

En caso de que la extensión de un archivo no diga cual es su tipo de MIME, se pueden definir las acciones a realizar con lo descrito en el archivo *MimeMagicFile*; mas si aún esto no se puede resolver, está una respuesta basándose en algún tipo de navegador, que se definen con la directiva *BrowseMatch*.

Las siguientes localidades establecen valores o parámetros para definir si se va a mostrar el estado y rendimiento del servidor, así como información del mismo, éstas se definen por medio de varias directivas entre una instrucción <location *directorio*> y se establece el acceso y que información se va a acceder por un host.

Para la configuración de Apache como un servidor proxy y de cache, se activa quitando los comentarios a los módulos de proxi y de caché.

En el servidor, se determina que es necesario para dar funcionalidad a nuestra aplicación cambiar varios parámetros de la sección principal los cuales son, el correo del

administrador del servidor, el nombre del servidor, para que sea atendido por un DNS, si se utiliza el nómbre canónico del servidor o la dirección IP y el lugar de donde se sirven las páginas o las aplicaciones.

3.3.3 Configuración de hosts virtuales

La tercera sección de la configuración de apache, es la configuración de hosts virtuales, cuando se requiere que el sitio sirva o responda a varios nombres de dominio y muestre páginas diferentes por cada host virtual que sea solicitado por el usuario o por el host que haga la petición. En sí, es mantener múltiples dominios y nombres de host en la misma máquina.

Para establecer un host virtual, se utiliza la directiva *NameVirtualHost*, y se configura por medio de la directiva *<VirtualHost>*, que establece el nombre del administrador, el documento base de donde servirá las páginas, el nombre del servidor, el directorio de errores, así como los registros personalizados.

Al final del archivo de configuración de apache, se encuentran las directivas del módulo webapp, para que interactúe con tomcat, servidor de páginas dinámicas (JSP).

En el servidor se configuró un solo host virtual ya que no se va a atender a varios nombres y si se desean atender varias aplicaciones Web, semanejan con Tomcat, se detemina el host virtual de Apache y se cjda de alta con Tomcat, para que sean el mismo configurandolo, para ver esto en detalle se recomienda el Anexo B5.

3.3.4 Configuración módulo WebApp.

Para que el servidor pueda resolver páginas JSP, que sean requeridas por el cliente, y para que sirva aplicaciones Web dinámicas, es necesario tener un servidor conjunto para que éste atienda las peticiones que impliquen compilación y creación de páginas nuevas. Para éste trabajo, se utiliza el servidor Jakarta Tomcat, que puede resolver páginas JSP.

Es recomendable, para evitar conflictos y tener dos servidores con nombres del mismo host, que trabajen sobre el mismo nombre de dominio a lo que es necesario montar Tomcat con

Apache.

Para que trabajen conjuntamente los servidores es necesario agregar a Apache un módulo que permita interactuar con Tomcat, éste módulo es el Webapp. La función de éste módulo es proporcionar a Apache la forma de identificar las peticiones que son de una aplicación Web (JSP) y que éste mande la petición a que sea resuelta por el servidor Tomcat y la respuesta generada, la conteste Apache.

Es necesario descargar los archivos necesarios para la configuración de éste módulo, los que son: Jakarta-Ant y Jakarta-Connectors.

Se descomprimen los conectores, se instala el Yakarta-Ant, se crean los módulos y se copia al directorio de apache en donde se encuentran los módulos el archivo que se llama *mod_webapp.so*, y por último, se configura el archivo httpd.conf, el cuál carga el módulo y redirecciona el puerto y determina el directorio base de las aplicaciones que se pretenden desplegar y que sean atendidas por éstos servidores Web.

Ésta configuración se puede obsrvar en el Anexo B.

3.4 Seguridad

En cuanto a la seguridad que se puede implementar en el servidor Web Apache, destaca el archivo .htacces, en donde se dan de alta los usuarios que van a poder accesar a ciertas carpetas que se encuentran en el contenedor. La implementación de la seguridad se hace por medio de los módulos de autentificación *(mod_auth)* y el módulo de acceso *(mod_acces)*, que interactúan con sus directivas, que se describen a continuación:

1. ***Allow***, permite especificar si es permitido el acceso
2. ***AuthGroupFile***, permite a un grupo determinado de usuarios accesar a los diferentes archivos dentro del sitio
3. ***AuthName***, especifica el tipo de restricción para los archivos, un ejemplo es "*Restricted Files*"
4. ***AuthType***, establece el nivel de restricción para dichos directorios, un por

ejemplo, "Basic"

5. ***AuthUserFile,*** establece la localización del archivo que contiene a los usuarios y sus contraseñas cifradas, éstas se generan por medio del comando htpasswd
6. ***Deny,*** trabaja como la directiva *allow*, que niega o da acceso al servidor, esto puede ser a través de la IP o del nombre del host de la computadora que esta realizando la petición
7. ***Options,*** controla que características del servidor están disponibles en un directorio en particular
8. ***Require,*** por último, esta directiva establece el nombre del usuario requerido para un directorio o archivo específico.

3.4.1 Configuración de usuarios

Para dar de alta usuarios dentro de un servidor, se requiere un nivel de seguridad y control de acceso para el servidor. Para lo cual los usuarios tendrán privilegios de acceso para algunos directorios del servidor. La sintaxis para dar de alta un usuario es la siguiente:

```
htpasswd -c $APACHE_HOME/passwd/passwords USER_NAME.
```

Se tiene la opción de configurar un grupo para que varias personas tengan acceso a una zona específica del servidor, esto se hace por medio de la directiva *AuthGroupFile*, en donde en el directorio *passwd* de apache se especifica un archivo *Groups,* que determina a los usuarios que pertenecen a un grupo.

Un ejemplo de la configuración queda como sigue:

```
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /usr/local/apache/passwd/passwords
AuthGroupFile /usr/local/apache/passwd/groups
Require group GroupName
```

Se debe de tener en cuenta los navegadores que se están utilizando con el servidor y cómo es que tratan los datos, ya que algunos tienen algunos bugs y no permiten se lleve a cabo la configuración óptima para el acceso y autentificación de los usuarios.

Después de configurar las 3 secciones del servidor, dejando la primera con los valores predeterminados, la segunda con modificaciones para que se adecue con el administrador y el lugar de donde se sirve la aplicación o los archivos Web a atender y el nombre con el cual se identifica al servidor y la tercera sección determinando el host virtual que hace referencia al modulo agregado para redireccionar las peticiones JSP para el servidor Tomcat, queda listo para su utilización y el administrador es el responsable de su mantenimiento.

4. JAVA

La existencia en la actualidad de un lenguaje que pueda conjuntar una serie de capacidades en su paradigma de programación, que permita una versatilidad y una flexibilidad para diseño y construcción de aplicaciones que pueden ser útiles no solo en un entorno práctico e interno de una empresa, sino que englobe toda una gama de recursos y posibilidades para que esa aplicación sea vista y utilizada desde el exterior de una forma confiable (una aplicación distribuida).

Desde hace varios años, el lenguaje que se ha preparado para adaptar estas necesidades a la industria ha sido JAVA ya que gracias a sus desarrolladores, se ha convertido en un lenguaje que tiene una basta experiencia y una historia sustentada en lo que son las aplicaciones distribuidas e Internet.

JAVA nace en un ambiente de desarrollo tecnológico como un proyecto para el diseño del control de aparatos electrodomésticos; aunque en un principio se llamó Oak y su creador principal James Gosling, desafortunadamente, el proyecto no tuvo gran auge comercial y fue dejado de lado.

Un tiempo después se ve la posibilidad y potencialidad del lenguaje para ganar terreno en el área de Internet y así se crea y se presenta JAVA en agosto de 1995[Froufe, 1].

4.1 Características y ventajas

4.1.1 Simple

Java es un lenguaje que ofrece una manera simple de aprendizaje, ya que es un lenguaje con una sintaxis muy parecida a la de C++, pero no tiene las dificultades de éste como lo son el manejo de memoria con los apuntadores, las referencias de memoria, la definición de las estructuras, la definición de macros y la necesidad de liberar memoria. Por esto JAVA ofrece un modelo simple de aprendizaje para introducirse al entorno y un fácil manejo de la programación.

4.1.2 Orientado a objetos

JAVA retoma el concepto de programación orientada a objetos, que nos determina el uso de clases, para la definición de objetos e instancias para las copias de estos objetos. Soporta las características del paradigma como lo es la encapsulación, la herencia simple y múltiple a través de las interfaces y el polimorfismo.

4.1.3 Distribuido

JAVA tiene diferentes herramientas y librerías para que los programas resultantes sean aplicaciones distribuidas y para la interconexión TCP/IP, ya que tiene rutinas para interactuar con los diferentes protocolos de red como lo son *http y ftp*.

4.1.4 Robusto

JAVA realiza verificaciones en tiempo de compilación así como en tiempo de ejecución de la aplicación, para que salgan todos los posibles errores en el desarrollo de la aplicación, además JAVA maneja la asignación y liberación de memoria para que el programador no se preocupe por los detalles de la recolección de basura y asignación de memoria dinámica además que realiza una verificación de los códigos de bytes (ByteCodes), que son el producto de la compilación para que pueda ser interpretado por la JVM.

4.1.5 Arquitectura neutral

JAVA compila su código a un archivo de texto independiente de la arquitectura de la computadora en que se ejecutará, ya que con la JVM, se puede ejecutar el código en cualquier máquina, independientemente de su Sistema Operativo.

4.1.6 Seguro

La seguridad en JAVA se divide en dos puntos principales, el primero es en relación con la seguridad de la programación y el código, que al ser compilado y antes de ejecutarse, se revisa que el archivo objeto (ByteCodes) para que no tenga ningún método o clase que intente acceder a una clase privada del sistema o ver si no existe un método que viole los derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto.

Y en cuestión de la parte de acceso, tiene un cargador que se encarga de separar las

variables de entorno del sistema de archivos de los recursos procedentes de la red.

4.1.7 Portable

JAVA implementa estándares de portabilidad, como por ejemplo que los enteros son siempre enteros y son de 32 bits de tamaño en complemento a 2. Otra característica, es que construye sus interfaces de usuario a través de un sistema abstracto de ventanas para que sean fácilmente implementadas en sistemas como UNIX, Mac o Pc.

4.1.8 Interpretado

El código de JAVA es un código que es Interpretado, no compilado como C++, siendo así que no hay que esperar al compilador por el código ejecutable, sino que el intérprete toma el código y lo va ejecutando línea por línea, aunque esto parezca ser lento ya en la actualidad, se puede comparar al interprete de JAVA con cualquier compilador del mercado y la diferencia es mínima o nula.

4.1.9 Multitarea

JAVA permite el desarrollo de procesos independientes dentro de procesos más complejos, para dar paso a hilos y se pueden estar ejecutando varios a la vez, para la rapidez de una aplicación, aunque están atados a las características de desempeño de la computadora en la cual se esta ejecutando la aplicación.

4.1.10 Dinámico

El dinamismo en JAVA se refiere a que el lenguaje va cambiando conforme avanza la tecnología, pero eso no implica que si hay que actualizar librerías o APIS, se tendrá que actualizar la aplicación que ya se tiene en funcionamiento y no hay que actualizarla a menos que así se requiera.

4.2 Apis de Java

En la Tabla 4.0 se muestran algunas de las APIs de JAVA para tener un panorama más claro de lo que se puede desarrollar con éste lenguaje y hasta donde tiene su alcance.

Tabla 4.0 APIS de JAVA	
API	*DESCRIPCIÓN*
Java Enterprise	Conjunto de especificaciones para entornos empresariales.
JDBC API	Java Database Connectivity, para permitir a las aplicaciones acceder a bases de datos en forma homogénea por medio del lenguaje SQL.
Java RMI	Remote Method Invocation, Llamadas a procedimientos remotos, para aplicaciones distribuidas.
JAVA IDL	Para trabajar aplicaciones distribuidas a través de CORBA.
JNDI	Java Namming and Directory Interface, permite que un servicio de directorios y de localización de recursos en una aplicación empresarial.
JavaBeans	Especificación de componentes basados en JAVA.
JAF	JavaBeans Activation Framework, API para la interaccion con los componentes JavaBeans.
Java Security API	Se encarga de los componentes que necesiten encriptación, llaves de seguridad, certificación, firmas y autentificación.
JFC	Java Foundation Classes, Jerarquia de clases para el desarrollo de aplicaciones con interfaces gráficas.
JFC- Swing	Conjunto de componentes gráficos para el desarrollo de interfaces gráficas.
JFC-Java 2D	Extensión de AWT, para el tratamiento de información gráfica bidimensional.
Java Servlet API	Crear applets que se ejecutan en el servidor.
Java Server API	Permite el intercambio de información entre un servidor web y alguna aplicacion en su entorno.
Java Commerce API	API para transacciones comerciales por Internet.
Java Media API	Conjunto de especificaciones para el acceso y utilización de información interactiva.
JMF	Java Media Framework, conjunto de especificaciones para la arquitectura, protocolos e interfaces de programación para reproductores multimedia, captura y video conferencia.
Java_Collaboration	Especificación para la comunicación interactiva bidireccional.
Java Telephony	Especificaciones para aplicaciones telefónicas.
Java Speech	Proporciona especificaciones para el reconocimiento y sintesis de la voz.
Java Animation	Permite la manipulación y movimiento para objetos bidimensionales.
Java 3D	Especificación para la manipulación de objetos tridimensionales.
Java Management API	Permite la gestión remota en redes.
Java Bluetooth API	Especificaciones que permitan integrar cualquier dispositivo a bluetooth.

La clasificación del lenguaje en este trabajo se conforma de seis subdivisiones, que son: Estándar, GUI, Red, Distribuido, Internet y Base de Datos que serán destacadas puntualmente en el desarrollo del capítulo, se muestra una descripción de cada una de ellas y abarca el área de Internet y Base de Datos, ya que este trabajo se basa en Aplicaciones en Internet, con interacción con Bases de Datos.

4.3 Clasificación del lenguaje JAVA

Las áreas en la división de las diferentes presentaciones del lenguaje de programación se destacan en los puntos subsecuentes, en la Figura 3 se describe el modelo propuesto para la clasificación del lenguaje según el funcionamiento de sus Api's.

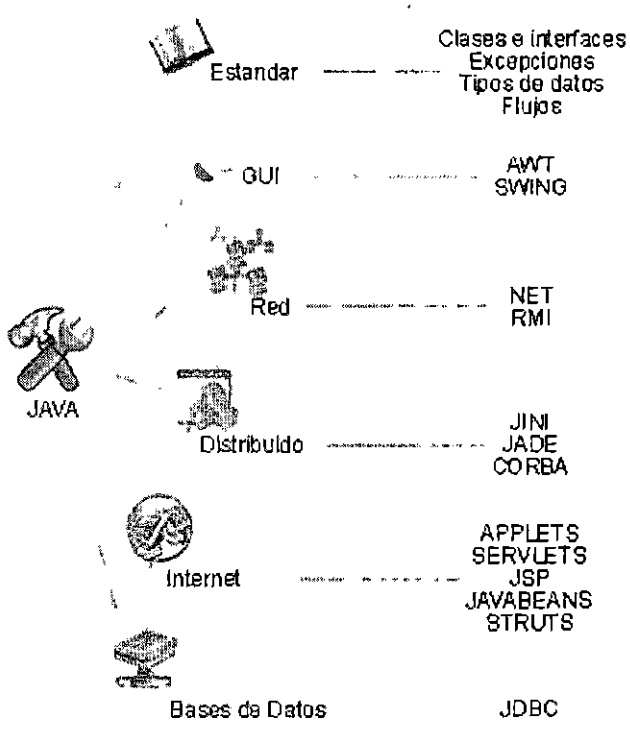

Figura 3. Clasificación de JAVA.

4.3.1 Estándar

La categoría ESTÁNDAR contiene las clases e interfaces, que es la descripción del leguaje en sí, es la forma básica de programar en JAVA aplicaciones para entrada y salida en modo de comandos.

Esta categoría especifica el manejo de las excepciones, o como es que se deben de implementar, ya que contiene por medio del paquete las funciones necesarias para la funcionalidad de las aplicaciones por medio del manejo de excepciones.

La categoría estándar contiene los tipos de datos que maneja JAVA, que son los tipos de datos básicos y los compuestos.

La última parte que se anexa a este primer módulo de clasificación son los flujos.

Clases e interfaces
En esta sección, se determinan las clases, que son tipos de objetos definidos por el usuario. Que se refiere a una abstracción de un objeto que será utilizado con base a las necesidades que se requiera. Las clases se componen de la definición de sus Propiedades y de sus Métodos.

Las propiedades o atributos, son los tipos de datos o las variables que son propiedad de la clase y que tendrá cada instancia que se cree de la misma.

Los Métodos son las funciones que servirán para interactuar con la clase accediendo y manipulando los atributos, para revisar el comportamiento de las instancias que se crean de la clase.

Excepciones
Las excepciones son las clases que nos permiten manejar cualquier evento que no se pueda atender por ningún procedimiento definido o implícito en el flujo de ejecución del programa.

Tipos de datos
JAVA Maneja ocho tipos de datos primitivos, que son los tipos de datos básicos para poder crear, manipular y eliminar las variables que se estén manejando en una clase.
Se manejan en el lenguaje cinco tipos de datos enteros que son el **byte**, **short**, **int**, **long** y **char**; los tipos reales que son el **long** y el **double** y el último es el **boolean**, que da sus valores **true** y **false**.

Existen otros tres tipos de datos que se manejan en el lenguaje, que son las **clases,** las **interfaces** y los **arrays.**

Las clases son estructuras o tipos de datos abstractos que contienen variables y métodos para su manejo, lo cual facilita la interacción con la misma y puede ser adaptable, algunas de las clases que son tipos de datos en Java son los Strings, Integers, Float, entre otros.

Las interfaces son las clases por las que se sirve JAVA par implementar la herencia múltiple, ya que el lenguaje no soporta la herencia múltiple y se implementa por medio de las interfaces, que son clases que contienen métodos abstractos, que se implementan en las clases en las cuales se van a utilizar esos métodos sobrescribiéndolos.
Los arrays se manejan en el lenguaje como clases que son colecciones que van a almacenar tipos de datos o incluso objetos, ya sean o no instancias de la misma clase, los cuales se recuperan por el método cast.

Flujos
En JAVA se manejan los Flujos que son de entrada y de salida, ya sea por medio de la consola, interfaces gráficas, archivos, peticiones http, etc.

4.3.2 GUI

Las Interfaces gráficas de JAVA se dividen en dos paquetes, la libreria AWT, Advanced Window Tool y la SWING que es una versión de componentes más especializados con una extensión de librerías para el manejo de gráficos para aplicaciones de escritorio personales o aplicaciones empresariales con una interfaz gráfica para los usuarios finales.

AWT
Este paquete maneja una gama específica de componentes gráficos como lo son los frames, botones, campos de texto, entre otros y contiene los manejadores de eventos para cada componente que exista en este paquete y en la librería SWING.

Swing
EL paquete SWING es una extensión de la librería AWT, ya que contiene elementos mejorados o con características específicas que los componentes de la librería AWT, se

caracterizan los componentes, incluso por la forma de declararlos, algunos son: JButton (Botones), Jframe (Formas), JtextField (Campo de texto), se caracterizan por la J que les antecede en la declaración del componente.

4.3.3 Red

La comunicación entre computadoras a través de un medio ya sea físico o inalámbrico se denomina una RED, para el intercambio de datos es necesario que se lleven a cabo una serie de lineamientos y protocolos para el tráfico de la información, JAVA, trata la parte de protocolos, encapsulamiento y manejo de la comunicación o transmisión de datos en la red a bajo nivel, dejando así al programador una tarea sencilla al momento de la programación a través de un sistema distribuido. Por otro lado JAVA soporta tener varias conexiones a la vez.

NET

Para el manejo de las redes, JAVA cuenta con herramientas bien definidas, como lo son los Sockets, que se manejan en sus múltiples formas en que JAVA los utiliza para las comunicaciones de red y los sockets Stream, son un modelo de comunicaciones orientado a la conexión, se basan en el protocolo TCP.

Los DatagramSocket Soportan la comunicación sin estar orientados a conexión, se basan en el protocolo de red UDP, esto se refiere a que los paquetes se envían por distintas vías, se pueden reenviar cuantas veces sea necesario, pero pierde un poco de garantía en cuestión del manejo de paquetes.

RMI

Una de las características de JAVA es que soporta RMI, Invocación de Métodos Remotos, esto se refiere a que un objeto instanciado en una JVM, puede acceder a los métodos de un objeto instanciado en una JVM diferente, independientemente si las JVM se encuentran o no en el mismo lugar. Esto significa que pueden estar en una red como lo es Internet y algún equipo cliente invoca un método remoto en un equipo servidor que se encuentra en otro lugar del mundo, esto por medio de la serialización que permite transportar objetos en paquetes a través de una red en forma de bits o mejor dicho código de bytes. Esto se hace por medio de una clase que se da de alta y se publica por un servidor que la instancia con

los métodos que pueden ser invocados desde otra clase en otra computadora.

4.3.4 Distribuido

En JAVA se establecen diferentes tecnologías para desarrollar sistemas distribuidos entre los que se destacan:

JINI
JINI es una tecnología especialmente diseñada para trabajar en ambientes distribuidos, permitiendo que cualquier dispositivo pueda agregarse a una red sin la necesidad de configurarse ni buscar algún protocolo por parte del usuario o programador, sino que JINI se encargue de todo lo que es configuración y el usuario llegue, conecte su dispositivo a la red y lo utilice inmediatamente.

JINI introduce un concepto simple: Los dispositivos deben de trabajar en conjunto [SUN, 2].

Establece cuatro principios de la programación distribuida, los cuales son:

Instant On; establece que los dispositivos son "plug", se conectan a una red e inmediatamente deben de funcionar estando disponibles sus recursos y servicios.

Impromptu community; el concepto abarca la creación de comunidades de dispositivos, que se pueden conectar sin ningún problema, dando capacidades a las redes de ser portables e incluso adherirse a otras comunidades JINI.

Resilient; Esto implica que la tecnología JINI se adapta a los cambios muy rápidamente; esto es, los usuarios van y vienen, pero la comunidad sigue.

Special delivery; Esto es que los servicios de JINI están disponibles con base a la demanda, están siempre que se necesiten [SUN, 2].

JADE
Java Agent Development es un middle-ware que permite la fácil implementación y

desarrollo de Agentes para realizar diferentes tareas en ambientes distribuidos, dando así la facilidad de la administración de procesos con base a agentes y el sistema de JADE se encarga de las comunicaciones de los mensajes y coordinación de los Agentes.

CORBA
Common Object Request Broker Architecture, es la implementación para procesos distribuidos en ambientes heterogéneos por medio de un IDL, JAVA tiene una implementación para poder desarrollar este tipo de interfaces trabajando la comunicación ORB por medio de RMI e IDLJ.

4.3.5 Internet

El auge que JAVA ha ganado en el mercado se debe a sus ya mencionadas características y por otra parte es el uso que se le ha dado a través de la Web, ya que en la actualidad muchas de las aplicaciones que se tienen en Internet por parte de empresas que requieren software específico distribuido por Internet, recurren a JAVA, desarrollando aplicaciones en las diferentes presentaciones y enfoques que tiene JAVA para aplicaciones en Internet o combinándolas para crear aplicaciones de Internet robustas y que permiten una interoperabilidad basada en el modelo cliente-servidor.

Esto se debe a que Java facilita la comunicación entre sistemas distribuidos y permite que sus clases sean serializables, esto es que se transporten en códigos de bytes y al recibirlas otra JVM se puedan utilizar sin ningún problema.

Applets
Los applets son elementos gráficos que se utilizan para la creación de aplicaciones que contengan una interfaz de usuario amigable, su auge se ha dado porque se pueden programar elementos de animación, movimiento y utilidades que se pueden agregar como un objeto a la mayoría de los navegadores de Internet que están en el mercado.

Éstos extienden de la clase applet y tienen un proceso determinado, ya que hay que inicializarlos, determinar su método principal y destruirlos después de haberlos utilizados para liberar los recursos del sistema que los está utilizando.

Los applets son aplicaciones pequeñas para que no sean pesadas para cargarse en un navegador y sean transportables, permiten la inclusión de sonido y animaciones, permitiendo abrir un canal de comunicación a través de un sistema distribuido por medio del protocolo de Internet.

Un applet que despliegue "Hola Mundo" quedaría como se ve en La siguiente figura.

```
import java.awt.Graphics;
    import java.applet.Applet;

    public class HolaMundo extends Applet {
        public void paint( Graphics g ) {
            g.drawString( "Hola Mundo!",25,25 ) ;
            }
        }
```

Figura 4. Estructura de un Applet

Un applet tiene un ciclo de vida definido, el cual consiste en:

1. Se crea una instancia de la clase que controla el Applet.
2. El applet se inicializa.
3. El applet comienza a ejecutarse.
4. El applet recibe llamadas de sus métodos, la primera *init()*, después *start()*, seguido de *paint()*; después todas las llamadas pueden ser recibidas asíncronamente[Froufe, 3].
5. Al final, el applet debe de detenerse y destruirse para que no siga consumiendo recursos del sistema.

Servlets

En cuestión de aplicaciones de Internet hay que tener en cuenta que los servicios que alguna empresa presta a través de un portal de Internet debe sujetarse a un esquema que proteja a la parte de la empresa, en este caso el servidor y por otra parte al cliente, para esto java propone un modelo de programación en donde la lógica de programación se lleve a cabo por servlets para que tengan el control de la aplicación en la parte del servidor y por

otro lado el cliente sólo mande datos a procesar y reciba sólo JSP's para visualizar una página en el navegador.

La interacción con el cliente a través de una aplicación Web por parte del servlet extiende de dos métodos, el *ServletRequest* y el *ServletResponse*, el primero establece los parámetros de entrada que va a recibir el servlet y el segundo manda la respuesta que generó el servlet[Palos, 4].

Un ejemplo de un Servlet es el que se muestra en las siguientes líneas de código de la Figura 5.

```
public class HolaMundoServlet extends HttpServlet
    {
        public void doGet (HttpServletRequest request,
                           HttpServletResponse response)
        throws ServletException, IOException
        {
        PrintWriter        out;
        String     title = "Hola Mundo";

            response.setContentType("text/html");
               out = response.getWriter();
            out.println("<HTML><HEAD><TITLE>");
        out.println(title);
        out.println("</TITLE></HEAD><BODY>");
        out.println("<H1>" + title + "</H1>");
        out.println("</BODY></HTML>");
        out.close();
        }
    }
```

Figura 5. Estructura de un Servlet

Java server pages

Java permite varias interfaces para el manejo de información a través de Internet, una de ellas es la tecnología JSP (Java Server Pages) que es una extensión de la tecnología servlet y permite la inclusión de código JSP y HTML, para que se puedan crear interfaces de

usuario sencillas y sin peso para un navegador de Internet, que al ser compiladas e interpretadas por el servidor, da como resultado un servlet, el cual da las ventajas y características ya descritas de los mismos. Esto da al programador un modo de programación simple y sencilla con base a etiquetas.

Entre las ventajas se destacan de la utilización de JSP se tienen:

- Utilizar la tecnología JSP sin aprender el lenguaje JAVA.
- Extender a JSP, por parte de un desarrollador de JAVA, puede migrar fácilmente a JSP.
- Fácil para escribir y dar mantenimiento a páginas por medio de sus librerías y etiquetas[SUN, 5].

Un ejemplo de una página JSP se muestra a continuación en la figura 6.

```
<%@ page contentType="text/html; charset=iso-8859-1"
language="java" import="java.sql.*" errorPage="" %>
<html>
 <head>
  <title>Hola Mundo</title>
 </head>
<body>
 <%
  out.print("<H1>Hola Mundo!!!</H1>");
 %>
</body>
</html>
```

Figura 6. Estructura de un JSP.

La compilación de los archivos JSP se realiza por medio de un servidor de páginas dinámicas, en éste trabajo se utiliza un servidor Tomcat una extensión de la Fundación Apache Software, éste servidor, recibe una petición JSP y la compila para dar como resultado un servlet que hace las peticiones y las respuestas a los navegadores que solicitan la página.

La estructura de almacenamiento del servidor se menciona más adelante dentro de éste documento, sólo se destaca que los contenedores de las aplicaciones y de las páginas dinámicas, deben adecuarse a dos directorios principales de la aplicación, por una parte el archivo web-inf, que contiene la información de la aplicación y las clases que se utilizan y por otro lado, el archivo Server.xml contiene la configuración para que se despliegien los componentes de la aplicación y que ésta sea accesible.

JavaBeans

Un JavaBean es un componente de Java, que consiste en clases y métodos generales que pueden implementar módulos de acción para diferentes comportamientos; estos componentes permiten ser retomados en cualquier aplicación en los que se requiera de alguno de sus métodos permitiéndoles independencia y serialización, lo que mantiene una de las ventajas de Java, programarlo una vez y ocuparlo en donde sea.

Cada componente tiene sus atributos y métodos, su característica es que por cada atributo se tienen dos métodos *getAtributo()* y *setAtributo()*, el primero es la interfaz para obtener el estado del atributo y el segundo es para asignar valor al atributo.

Struts

Struts es una herramienta de soporte para el desarrollo de Aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE, su característica de software libre y su compatibilidad con las plataformas de java lo hacen una herramienta fácil de utilizar y de implementar[Wikipedia, 8].

Por medio de struts se divide el desarrollo de la aplicación en tres partes, la parte de Modelo, que corresponde a la lógica del negocio la cual se comunica con la aplicación Web, el modelo es la interacción con la base de datos. La segunda es la vista, que corresponde a la apariencia de cómo se le presenta la información al usuario y recopila la información del usuario hacia la aplicación, struts cuenta con varias herramientas que proporcionan plantillas que permiten facilitar el diseño de tales plantillas.

El control es la parte que corresponde al flujo de desarrollo del sistema, es el que recibe los datos de entrada y la selección de un modelo para ser llamado, el componente del modelo

envía al control el resultado de los eventos y procedimientos para poder seguir con la aplicación.

Entre las características de Struts se identifican:

- Configuración centralizada.
- Interrelaciones entre páginas y acciones.
- Componentes de aplicación.
- Librerías de entidades.
- Herramientas de validación de campos.

4.3.6 Base de datos

JAVA permite la implementación de aplicaciones que interactúen con Bases de Datos, ya que tienen una API para que se puedan crear conexiones con una base de datos e interactuar por medio de SQL, a continuación se describe el API de JAVA para el manejo de bases de datos.

JDBC

La arquitectura de la API se divide en 2 rubros, la del controlador directo, que es una interfaz directa del controlador de la base de datos con JAVA, para que se realicen las consultas necesarias, ésta es una forma directa de interactuar con bases de datos. Por otro lado, se tiene la forma indirecta de interactuar con bases de datos, que es a través de un puente de comunicaciones entre las consultas de la aplicación enviadas por JAVA y el controlador de la base de datos.

El API de JAVA permite hacer tres cosas:

- Establecer una conexión con una base de datos o acceder a cualquier sistema tabular de datos.
- Mandar sentencias SQL.
- Procesar resultados.

Entre las características clave del API se destacan:

- Acceso completo a MetaDatos.

- Sin instalaciones.
- Conexión a bases de datos haciendo referencia por medio de la URL[SUN, 6].

Para el desarrollo de este trabajo se utilizará al lenguaje de programación JAVA, por su versatilidad de uso y aplicación en las diferentes plataformas que existen en el mercado hoy en día. Aunado a eso, se toma en cuenta que JAVA es un lenguaje portátil y fácil de programar en sus diversas formas, ya sea aplicaciones en Internet, Aplicaciones locales o para darle una versatilidad, con los Java Beans o los Enterprise Java Beans, que son formas de Aplicar el lenguaje de programación a necesidades que surjan en un campo de desarrollo, y se discutirán en detalle más adelante en este trabajo.

5. SERVIDOR DE PÁGINAS DINÁMICAS JAKARTA-TOMCAT.

Una página dinámica es uno de los componentes que conforman un sitio Web, la característica de éstas es que se generan con un lenguaje de programación robusto y código HTML en tiempo de ejecución, en base a la petición lanzada por el usuario que navega en Internet.

Para que en las páginas dinámicas se muestre cierta información, se deben de realizar una o más operaciones. Se puede decir que la petición del navegador manda llamar una rutina externa que está en el servidor y ésta rutina sólo coloca el resultado de la operación en la página Web como código simple de HTML.

Para éste trabajo se emplea JAVA como lenguaje de programación robusto; esto conlleva a un servidor que pueda compilar éste tipo de páginas, la Fundación Apache y Sun MicroSystem, crearon el proyecto de Jakarta Org, que desarrolló Tomcat, un servidor de páginas dinámicas que compila directivas o código de Java incrustado en una página HTML, mejor conocidas como páginas programadas en JSP (JAVA Server Pages).

En vista de la necesidad de nuevas aplicaciones Web que proporcionen dinamismo al momento de generar las páginas que presenta un servidor, la Fundación Apache, extiende a un proyecto de un contenedor de páginas que se generen al momento de recibir la petición, lo cual por el auge de JAVA se toma como generador de dichas páginas por medio de su presentación de Servlets y JSP, así nace Jakarta-Tomcat, una extensión de Apache para servir páginas dinámicas JSP.

La seguridad en una aplicación Web es importante y no se puede dejar que todos los visitantes de un sitio puedan ver cualquier información confidencial o el código de cómo se realizan algunas operaciones dentro de un programa en una página Web; por eso es necesario hacer que el control de ciertas operaciones se realicen en el servidor, generando

una respuesta para el navegador que sea solamente HTML. Esto se retoma del paradigma de Java que se refiere al Modelo-Vista-Controlador, lo cual establece que los servlets mantengan el control de la aplicación y el flujo de la información, las páginas JSP y HTML, la vista, que es el diseño al usuario y el control se maneje a través de las interfaces y eventos que se den dentro de la aplicación.

5.1 Características.

Las características del servidor Web Tomcat es que trabaja con JAVA, esto es que no depende de plataforma alguna, ya que en la compilación se generan códigos de byte, que son interpretados por la JVM. Otra característica importante es que sus archivos de configuración son de texto y se pueden editar en cualquier programa de edición de textos, otra es su distribución libre y no tiene costo alguno y se puede descargar de la página de Apache. En el servidor se pueden delegar permisos de acceso a determinadas carpetas y excluirlas del directorio que queda a la vista y disposición de visitantes del sitio. Tomcat, con la ayuda de la máquina virtual, es un generador de páginas de Internet, que se basan en las entradas proporcionadas por el usuario.

Otra característica de Tomcat es que no sólo procesa páginas en JSP, sino también muestra páginas en HTML y puede manejar Servlets y JavaBeans. Tomcat se puede configurar para que se escuche a un puerto lógico determinado o que colabore con Apache por medio de conectores.

De este modo, Tomcat atiende las páginas en JSP y apache procesa todo lo que sea HTML, esto hace más funcional un sitio, porque no se tendrían dos servidores independientes para atender los diferentes tipos de páginas que se manejen, sino que con un mismo servidor se pueden atender páginas que tengan diversos códigos, ya que se resuelven códigos en HTML, JSP y Servlets; además apache contiene un módulo para procesar PHP y CGI's.

Tomcat da una versatilidad importante en cuanto a aplicaciones en Internet para cualquier ramo, ya que java tiene muchas ventajas para el tratamiento de información en Internet, que es lo que ha puesto al lenguaje en la punta para este tipo de aplicaciones.

5.2 Ventajas

Las ventajas del servidor Web Tomcat son variadas, pero simples:

- Su costo; es gratuito, se puede descargar de la página de Apache.
- Es independiente de plataformas, ya que lo único que necesita es la JVM.
- Puede ejecutar varios tipos de subrutinas, ya sean Servlets, Beans o JSP.
- Se puede configurar para que sea o no independiente de otro servidor.
- Evita la copia de código, por medio de la utilización de clases, ya que se cargan de los archivos .class ya compilados en directorios aparte de donde se almacenan las páginas Web.
- Facilita la administración de un sitio, ya que no hay que tener una página Web por cada entrada de datos, sino que se genere la página con base a la información proporcionada por el usuario por medio de algún formulario.

5.3 Administración

Para la administración del Servidor Tomcat, es importante conocer sus directorios, ya que estos son la parte fundamental del Servidor, los directorios de Tomcat, tienen cierto parecido a los directorios de Apache, ya que este cuenta con un directorio específico para la configuración del servidor, un directorio para guardar los registros que se determinen, un directorio en donde se encuentran las librerías de donde se van a obtener los archivos class, y las páginas precompiladas para que no se tengan que compilar cada vez que se accede a ellas.

Los archivos que determinan el funcionamiento en el servidor, se instalan en un sólo directorio, lo cual no es definitivo ya que se puede configurar para que los archivos de configuración no estén en el mismo directorio en donde se encuentran los archivos que van a estar accesibles para los diferentes visitantes del sitio Web.

En la Tabla 5.0 se muestran los directorios de configuración de Tomcat:

Tabla 5.0 Directorios de configuración Tomcat			
NOMBRE	DESCRIPCIÓN	TIPO	RUTA
bin	Contiene los comandos de inicio y fin de Tomcat.	Archivo	/usr/local/tomcat
conf	Contiene los archivos de configuración del servidor como el server.xml	Archivo	/usr/local/tomcat
logs	Es en donde se almacenan los archivos de registro del servidor.	Archivo	/usr/local/tomcat
Lib	Contiene los archivos .jar, que son clases ya definidas para su utilización (Librerías).	Archivo	/usr/local/tomeat/common
Classes	Contiene clases que se utilizan en alguna aplicación y tendrán un lugar en el classpath.	Archivo	t/common
Webapps	Es el archivo que contiene las páginas Web hechas con JSP y de donde tomcat las sirve.	Archivo	/usr/local/tomcat
Work	Es en donde Tomcat coloca páginas intermedias o ya compiladas, para que cuando se solicite de nuevo ya no las compile de nuevo.	Archivo	/usr/local/tomcat

La configuración principal del servidor, se encuentra en el archivo llamado Server.xml, que esta dentro del directorio conf, en este archivo, se puede llevar a cabo la administración del servidor, aquí como en apache, es en donde se incluyen las directivas para llevar a cabo la administración de directorios, usuarios y seguridad.

El archivo Server.xml tiene una sintaxis parecida a HTML, ya que se programa por medio de etiquetas.

Se debe de tener en cuenta varias consideraciones básicas para el funcionamiento del servidor, así como el total conocimiento de las partes que lo componen, el archivo de configuración de Tomcat está compuesto por diferentes directivas, que engloban la parte

funcional del mismo, el elemento principal, Server, que se encarga de todo el funcionamiento del servidor, ya que dentro de ésta directiva se manejan todas las directivas, la que le sigue en importancia es la directiva service, que representa a todo un conjunto de Conectores que son enlazados a un Engine. Un Conector representa la interface de un cliente externo que manda una petición a un Servicio en específico y se encarga de recibirlas.

Un Contenedor son los componentes que tienen como función principal procesar las peticiones entrantes y crear la respuesta correspondiente; así un Engine maneja todas las peticiones para un determinado Servicio, un Host, maneja las peticiones entrantes dirigidas a un host virtual y un Context, se encarga de los procesos correspondientes para el manejo de las peticiones para determinadas aplicaciones Web.

El servidor cuenta con Componentes anidados, los cuales se declaran dentro de un Contenedor y algunos dentro de un Contexto.

Por cada elemento que se maneja dentro del archivo de configuración del servidor, se maneja una estructura esencial, la cual se describe como:

1. ***Introducción:*** Es la especificación del componente, que se forma de una Interfaz de Java, que es implementada por una o más implementaciones estándar, su estructura es (org.apache.catalinaPAQUETE).
2. ***Atributos:*** Son el conjunto de datos o atributos que son soportados por el elemento. Generalmente se van a dividir en Atributos Comunes que son soportados por todas las implementaciones de las interfaces de Java, y los atributos de Implementación Estándar que son específicos de una clase de Java que implementa a su interfaz.
3. ***Componentes Anidados:*** Enumera cuáles de los componentes anidados pueden ser introducidos y soportados por el elemento en donde se están declarando.
4. ***Características Especiales:*** Describe la configuración de una gran cantidad

de características específicas para cada tipo de elemento que son soportadas por la implementación estándar de la interfaz.

Características de las directivas de Tomcat.

5.3.1 Server

El Server representa completamente al Contenedor de servlet Catalina, sin embargo, se debe de tener sólo un elemento de estos por servidor, en el archivo de configuración server.xml.

Como atributos comunes, se soportan los siguientes:

Tabla 5.1 Atributos comunes Server

ATRIBUTO	DESCRIPCIÓN
ClassName	Es la clase de Java que especifica la implementación a usar. Esta clase debe de implementar la interfaz org.apache.catalina.Server. Si no hay un nombre de clase definido, se utiliza una implementación estándar.
Puerto	El número de puerto TCP/IP por el cual el servidor recibe el comando para apagarse. Esta conexión debe de ser iniciada desde la computadora que corre la instancia de Tomcat.
Shutdown	El comando que debe de ser recibido por el puerto para que el servidor se apague.

La implementación estándar de Server es org.apache.catalina.core.StandardServer. Así se soporta el siguiente atributo:

Tabla 5.2 Atributo específico Server

ATRIBUTO	DESCRIPCIÓN
Debug	Es el nivel de debugueo o detalles a imprimir durante la ejecución de un programa, su parámetro es numérico y el nivel predeterminado es cero.

Los componentes que se pueden anidar dentro de esta etiqueta son:

1.Service. Uno o más elementos de servicios.
2.GlobalNamingResources. Configura el JNDI de los recursos globales del servidor.

Este elemento no contiene características especiales.

5.3.2 Service

Service es un conjunto de Conectores que comparten el mismo elemento *Engine* para el proceso de las peticiones entrantes. Uno o más componentes de esta directiva pueden estar anidados dentro de la directiva Server.

Como atributos comunes soportan los siguientes:

Tabla 5.3 Atributos comunes Service

ATRIBUTO	DESCRIPCION
className	Es la clase en Java de la implementación a utilizar. Esta clase debe de implementar la interface org.apache.catalina.Service. Si no se especifica se utiliza el valor predeterminado.
Name	Es el nombre a desplegar de este elemento, el cual se incluirá en los mensajes Log si se utiliza la configuración estándar de componentes Catalina. El nombre de cada Servicio asociado a un Server, debe de ser único.

La implementación estándar para un Service, es org.apache.catalina.core.StandardService, que soporta los siguientes atributos adicionales:

Tabla 5.4 Atributo específico Service

ATRIBUTO	DESCRIPCION
debug	Es el número que determina el nivel de debug para este componente. Si el número es más grande, se guardan más detalles del servicio en el archivo Log. Si no se especifica, el nivel es cero.

Los únicos elementos que se pueden anidar dentro de éste componente son los Conectores, seguidos inmediatamente de un Engine.

El elemento Service no tiene características especiales.

5.3.3 Conector (JTC)

JTC significa Jakarta-Tomcat-Connectors. Es el lugar en donde se encuentran los archivos fuente de los conectores. Éstos son liberados junto con Tomcat.

Existen dos tipos diferentes de conectores. Conectores que permiten a los navegadores conectarse directamente con Tomcat y los conectores que establecen conexión por medio de un servidor Web.

Los conectores que permiten las conexiones directas se encuentran en archivos binarios de Tomcat, archivos .jar.

Los conectores que se utilizan en conjunto con un servidor Web se crean de dos componentes. Uno escrito en Java y otro escrito en C. La parte de Java está compuesta de los archivos jar.

Algunos de los conectores principales de Tomcat, que utilizan una conexión directa, se muestran en la Tabla 5.5.

Tabla 5.5 Conexiones directas TOMCAT

CLIENTE	PROTOCOLO	NOMBRE DE LA CLASE EN TOMCAT	NOTAS
Navegadores Anteriores	HTTP/1.0	Org.apache.catalina.connector.http10.HttpConnector	Desaprobado
Navegadores Modernos	HTTP/1.1	Org.apache.catalina.connector.http.HttpConnector	Desaprobado
Navegadores Modernos	HTTP/1.1	Org.apache.coyote.tomcat4.CoyoteConnector	Actualmente en uso.

A continuación se muestran algunos de los conectores que realizan una conexión a través de un servidor Web:

Tabla 5.6 Conectores TOMCAT			
CLIENTE DEL SERVIDOR WEB	**PROTOCOLO**	**LADO DE TOMCAT**	**NOTAS**
mod_jserv	AJP/1.2	Ajp11	Obsoleto
mod_jk	AJP/1.3	CoyoteConnector con JkCoyoteHandler	Habilitado por default en 4.1; soportado en 4.0
mod_jk2	AJP/1.3	CoyoteConnector con JkCoyoteHandler	Es un conector actualmente desarrollado, está por default en 4.1
mod_webapp	WARP/1.0	WarpConnector	Soportado por Apache 2.0 y 1.3

Un servidor web puede albergar diferentes tipos de aplicaciones, programadas en varios lenguajes, como PERL, PHP, C o cualquier otro. Si una aplicación es escrita en Java utilizando la API Servlet, un conector es requerido para direccionar las peticiones del servidor Web al Servlet Engine. Dependiendo del servidor Web se requiere un conector específico. La mayoría de los servidores Web permiten la carga dinámica de extensiones (DLL). Algunas extensiones están disponibles como conectores en distribuciones binarias.

5.3.4 Conector Webapp

El conector Webapp representa un componente que comunica con un conector web por medio del protocolo WARP. Éste es utilizado cuando se requiere integrar invisiblemente Tomcat 4 con un servidor Web Apache y además si se requiere que apache maneje el contenido estático de la aplicación o que utilice procesamiento de SSL. En muchos entornos de aplicaciones, esto deriva en un mejor funcionamiento que correr las aplicaciones en el modo de stand-alone de tomcat utilizando el conector HTTP/1.1.

FIXME – es una descripción general de lo que pasa en el servidor al tiempo de que arranca cuando se tiene un Conector WARP configurado.

Los atributos que son soportados por los conectores son:

Tabla 5.7 Atributos conectores	
ATRIBUTO	DESCRIPCIÓN
className	El nombre de la clase de Java de la implementación a utilizar. Debe de implementar la interface org.apache.catalina.Connector.
enableLookups	Configurar con true si se requiere mandar llamar al método request.getRemoteHost() para que el DNS realice lookups para regresar el nombre del host del cliente remoto. Si está en false se regresa sólo la dirección fisica en un String. Por default está deshabilitado.
redirectPort	Si el servidor soporta peticiones non-SSL, y se recibe una petición como si fuera un <constraint de seguridad>, se requiere transportar como SSL, así Catalina redirige automáticamente la petición al puerto especificado en éste parámetro.
scheme	En este atributo se coloca el protocolo con el que se desea sean regresadas las llamadas del método request.getScheme(). Por ejemplo, se debe de colocar "https" para un conector SSL. El valor predeterminado es "http".
Seeure	Se utiliza para habilitar o deshabilitar la llamada al método request.isSecure(), y regrese a "true" para las peticiones de este conector. El valor predeterminado es falso.

5.3.5 Warp Engine

mod-webapp ofrece un *Engine* específico cuando se corre el Conector Webapp en su propio servicio. Este *Engine*, permite crear y mapear los hosts correspondientes al Host Virtual definido en el archivo de configuración de Apache httpd.conf.

La implementación estandar del Webapp Engine es org.apache.catalina.connector.warp.WarpEngine. Soporta el estándar Engine.

La característica especial de éste componente es FIXME que documenta cualquier característica especial soportada por el conector WARP.

5.3.6 Contenedor context

El elemento Context, representa una Aplicación Web, la cual corre en un determinado host virtual. Cada Aplicación Web se basa en un Archivo de Aplicación Web (WAR), o en una carpeta con el contenido correspondiente y desenpaquetado de las clases y páginas.

La aplicación que va a responder por cada petición HTTP, es seleccionada por Catalina, basándose en la comparación entre el prefijo de la petición URI y el context path de cada Context definido. Una vez que se elije el Context, éste selecciona el servlet apropiado para procesar las peticiones entrantes, de acuerdo con los mapas de servlets definidos en el archivo despliegue de la aplicación web (debe de estar localizado en /WEB-INF/web.xml).

Se pueden tener tantos Context como se requiera, anidados en un elemento Host dentro del archivo de configuración conf/server.xml. Cada Context debe de tener un único context path, el cual es definido por el atributo path. Adicionalmente, se debe de definir un Context con un context path vacío, para que éste se convierta en el Context predeterminado para alguna aplicación web en el host virtual, así cualquier petición que no concuerde con algún context path, sea atendido por éste Context predeterminado.

También, relativo a anidación de Context dentro de Hosts, se pueden declarar los Context y almacenarlos en archivos individuales con la extensión xml en el directorio appBase.

Los atributos comunes que soporta un componente Context son los que se muestran en la Tabla 5.8.

Tabla 5.8 Atributos comunes Context	
ATRIBUTO	**DESCRIPCIÓN**
className	Es el nombre de la clase de la implementación a utilizar. Debe de implementar la interfaz org.apache.catalina.Context. Si no se especifica se utiliza un valor predeterminado.
cookies	Se coloca como true por default, y es utilizado para que la identificación de comunicaciones por parte del eliente. Se coloca en false si se desea que el cliente teclee la URL cada vez que se quicra ingresar a la aplicación.
CrossContext	Colocar en verdadero si se desean llamadas de alguna otra aplieación con el método ServletContext.getContext() para crear un despachador de peticiones en un host virtual.
docBase	El documento base (también llamado como la raíz de contenido) de la aplicación web, si se está utilizando un archivo WAR es el path de en dónde se eneuentra éste archivo. Se debe de especificar un path absoluto para la aplicación del host propietario.
override	Colocar cn verdadero si se desea sobre escribir eiertos valores dentro del contexto del host contenedor de la aplieación.
privileged	Se utiliza para permitir a este contexto el manejo de servlets de administración.
Path	Es el context path por el cual se hace la comparación de la URL, cada path debe ser único para cada context. Si no se define nungún path, éste será el Context por default para este host.
reloadable	Este parámetro establece si Catalina va a monitorear las librerías de la aplicación y que automátieamente recompile la aplicación si han existido cambios.
wrapperClass	Es la clase de Java que implementa el servlet de administración para este Contexto, el valor es: org.apache.catalina.Wrapper.

La implementación por default del Context es org.apache.catalina.core.StandarContext. La cual soporta los atributos adicionales mostrados en la tabla 5.9.

Tabla 5.9 Atributos adicionales Context	
ATRIBUTO	**DESCRIPCIÓN**
debug	El nivel de detalle que se quiere almacenar al tiempo de ejecución de este elemento en los archivos Log.
swallowOutput	Este atributo permite valores en verdadero o falso para determinar si los mensajes System.out y System.err van a ser redireccionados a un archivo log. Si no se especifiea, el valor predeterminado es falso.
useNaming	Se requiere este valor para determinar si se permite a Catalina tener un Contexto JINDI inicial para la aplicación web, que es compatible con J2EE.
workDir	Es la ruta hacia el directorio que va a ser dispuesto temporalmente de lectura y escritura para los servlets que están en la aplicación. Si no se especifica, se toma el path de CATALINA_HOME/work.

Los componentes anidados que soporta éste elemento son:

1. ***Loader*** (Cargador): Configura el cargador de clases para la aplicación, que serán utilizadas por los servlets y beans.
2. ***Logger***: Configura el loger que manejará los mensajes que serán del Contexto.
3. ***Manager*** (Administrador): Configura la administración de las sesiones, de inicio, detención y la persistencia.
4. ***Realm*** (Dominio): Permite configurar un dominio para el uso de la base de datos para los usuarios y sus roles asociados para ser utilizados únicamente por una aplicación web en particular. Si no se especifica, se utilizará un dominio o permiso con base a el host propietario o el Engine.
5. ***Resources*** (Recursos): Configura la administración de los recursos que serán utilizados por la aplicación.

Éste componente define las siguientes características especiales:

1. ***Access Logas***: Este atributo permite la configuración para que Catalina escriba en un archivo específico los logs que se generen de una aplicación en particular.
2. ***Automátic Context Configuration***: Este parámetro permite que Catalina utilice

valores predeterminados para ciertos atributos que no sean declarados para cierta aplicación, como lo son, Loader, Manager, Resources.

3.***Context Parameters***: Se pueden definir valores de atributos que serán vistos por los servlets de la aplicación declarándolos entre los identificadores <param> y </param>, sin tener que agregarlas al archivo web.xml.

4.***Environment Entries***: Son nombres de variables que serán vistas por la aplicación para poder utilizarlas, por ejemplo el uso de cierta clase, así como su ubicación; esto se hace por medio de las etiquetas <Environment>, sin tener que agregarlas al archivo web.xml.

5.***Lifecycle Listeners***: Si se tiene una clase que necesite saber si el contexto está iniciado o detenido, se declara un escuchador (Listener), anidado en el contexto.

6.***Request Filtres***: Este componente nos permite configurar que un Context reciba peticiones de direcciones IP específicas, comparándolas con un archivo que determina los hosts permitidos y los hosts que no están permitidos.

7.***Resource Definitions***: Se pueden anidar recursos extra, que son ocupados por la aplicación cuando se configura para JNDI, por medio de ese contexto y no es necesario declararlos en el archivo web.xml.

8.***Resource Parameters***: Son parámetros anidados que configuran los recursos, como el "manager", cuando se utilliza JNDI.

9.***Resource Links***: Este elemento se utiliza para crear una liga a un elemento JNDI.

5.3.7 Engine

El elemento *Engine* representa el procesamiento entero del hardware para atender las peticiones de un servicio particular de Catalina. Recibe y procesa todas las peticiones de uno o más conectores, y regresa la respuesta completa para el Conector que la enviará al cliente.

Sólo debe de declararse un Engine anidado dentro de un elemento Service, seguido de su correspondiente conector asociado a este servicio.

Todas las implementaciones del Engine soportan los atributos de la Tabla 5.10:

Tabla 5.10 Atributos comunes Engine	
ATRIBUTO	**DESCRIPCIÓN**
className	Es el nombre de la clase que implementará el funcionamiento de éste componente. Implementa la interface org.apache.catalina.Engine.
defaultHost	Es el nombre del host que identifica las direcciones que se procesaran para las peticiones que son dirigidas a determinados hosts que no están en el archivo de configuración. El nombre debe de coincidir con uno de los elementos hosts anidados.
jvmRoute	Es el identificador que debe ser utilizado para la carga de escenarios para permitir una afinidad en las sesiones. El identificador debe de ser único entre todos los Tomcat que participan en un cluster, que generarán un identificador de sesión.
Name	Es el nombre que se utilizará para los mensajes de error para este Engine en los archivos Log.

La implementación estándar de *Engine* es org.apache.catalina.core.StandardEngine, que debe soportar el siguiente atributo adicional.

Tabla 5.11 Implementación estándar Engine	
ATRIBUTO	**DESCRIPCIÓN**
Debug	El nivel de almacenamiento de la información de debugs del Engine.

Se pueden anidar uno o más componentes dentro del elemento Engine, cada uno representando a un host virtual diferente asociado con el servidor.

Se puede opcionalmente anidar un Contexto predeterminado (DefaultContext), para definir las características de la aplicación que será desplegada automáticamente.

5.3.8 Host

El elemento Host, representa un Host virtual, el cual está asociado a un nombre de red en un servidor. Este nombre debe de estar dado de alta en un DNS, que administre el dominio de Internet al que pertenezca.

En algunas ocasiones, algunos administradores de sistemas, desean asociar más de un nombre de red con el mismo host virtual y aplicación. Esto se soporta por medio de la característica Host Name Alias de la siguiente forma:

Uno o más elementos Host son anidados dentro de un elemento Engine. Dentro del elemento Host se pueden anidar elementos Context para las aplicaciones Web, asociadas con un Host virtual. Cada elemento Host asociado a un Engine debe de tener un nombre que sea igual al atributo defaultHost del Engine.

Los atributos comunes que se soportan por éste elemento son los que se muestran en la Tabla 5.12.

Tabla 5.12 Atributos comunes Host

ATRIBUTOS	DESCRIPCIÓN
appBase	Es el directorio base de la aplicación, contiene la ruta o dirección en dónde se encuentra la aplicación Web que va a ser desplegada en el host virtual. Se debe de especificar un path absoluto para éste atributo o se puede generar una automáticamente por medio de algunas propiedades de Catalina como Automatic Application Deployment.
autoDeploy	Este es un indicador para verificar si la aplicación, se debe de desplegar automáticamente por el host configurator. El valor predeterminado es verdadero.
className	Es el nombre de la implementación a utilizar. Ésta clase debe de implementar la interface org.apache.catalina.Host.
Name	Es el nombre de red que se le da a éste host, que debe de estar dado de alta en el servidor de nombres de dominio.

La implementación estándar de un elemento Host es org.apache.catalina.core.StandardHost. Que soporta los atributos que se mencionan a continuación en la tabla 5.13.

Tabla 5.13 Implementación estándar Host	
ATRIBUTO	**DESCRIPCIÓN**
Debug	Es el nivel de debug que se desea para este elemento, ya que con base al número, se determina cuánta información se va a almacenar en los archivos Log. El nivel predeterminado es cero.
deployXML	Sirve para determinar si se va a desplegar la aplicación por medio de un archivo XML que utiliza un Context. También deshabilita la posibilidad de instalar directorios de aplicaciones o archivos war con el administrador de la aplicación que no están en el directorio base del host.
errorReportValveClass	Es el nombre de la clase que implementa el reporte de errores *valve*, que son los que se generan al desplegar una página que es atendida por Tomcat y éste *Valve* despliega los errores generados. Esta clase debe de implementar la interface org.apache.catalina.Valve.
liveDeploy	Este valor determina si una aplicación es lanzada, mientras Tomcat está corriendo, se despliegue automáticamente, este valor esta en verdadero por default.
unpackWARs	Este atributo define si las aplicaciones que están en archivos WAR, se desempaquetan a un directorio, o se despliegan directamente del paquete, sin desempacar a un directorio.
workDir	Es el nombre de la ruta en donde se encuentra un directorio temporal que es de lectura y eseritura, para los servlets que se encuentren en la aplicación mientras ésta esté corriendo. Si no se declara, se utiliza el directorio por default ubicado en $CATALINA_HOME/work.

Las características especiales que se soportan por éste componente son:

1. Access Logs. Cada vez que se corre un servidor web, normalmente se genera un archivo de acces log, el cual genera una línea de información para cada petición que es atendida por el servidor en un formato estándar.
2. Automatic Application Deployment. Si se está utilizando la configuración de Host estándar, se realizan las siguientes acciones cuando se arranca Catalina por primera vez.

a) Se asume que en el directorio definido como appBase se encuentra un archivo XML que contiene un CONTEXT para una determinada aplicación Web o para un archivo WAR.
b) Cualquier aplicación Web en WAR, que no haya correspondido al directorio del mismo nombre, se despliega automáticamente a menos que la propiedad de unpakWARs no esté activada. Si se re-despliega una aplicación porque es renovado el archivo WAR, se tiene que borrar el directorio del paquete anterior, para actualizar el nuevo paquete.
c) Cualquier directorio que tenga parecido al de una aplicación Web (un directorio WEB-INF/web.xml), va a recibir automáticamente su CONTEXT, aunque no se mencione en el archivo de configuración conf/server.conf.

El atributo *Host Name Aliases* se utiliza para que el mismo Host tenga diferentes nombres de dominio, que se resuelven en un DNS para que una misma dirección IP resuelva varias peticiones, por medio de estos diferentes nombres de dominio, éstos se configuran en un elemento separado de Host dentro del archivo de configuración server.xml.

Sin embargo, se puede tener que dos nombres de dominio diferentes accesen a una misma aplicación. Esto se resuelve por medio del elemento Alias anidado dentro del Host. Otro atributo es el del elemento Lifecycle Listeners, el cual va a notificar el estado en el que se encuentra el servidor.

La característica Request Filters, permite establecer los Hosts que tendrán acceso al servidor y aquellos que tienen el acceso restringido.

Otra característica especial es la de Single Sign On, que se utiliza para cuando es necesario que en una aplicación, un usuario se autentifique sólo una vez mientras tenga desplegada la aplicación.

Esta característica, opera bajo las siguientes reglas:

- Todas las aplicaciones Web configuradas para éste Host virtual comparten el mismo dominio. Esto significa que se anidará un Realm dentro del elemento Host, pero no dentro del Context.
- Mientras el usuario sólo accese a recursos sin protección, no se pedirá que se autentifique.
- En cuanto el usuario accese a una zona protegida, se le pedirá que se autentifique por medio del método definido por la aplicación misma.
- Una vez que el usuario se autentifica, todas las aplicaciones relacionadas con el control de acceso, serán notificadas para que el usuario no tenga que autentificarse cada que accesa a cada una de ellas.
- Una vez que el usuario sale de la aplicación o que se le termina tiempo de sesión, se invalida el acceso para todas las aplicaciones relacionadas; y si desea ingresar de nuevo, se pedirá que se autentifique de nuevo.
- Esta característica funciona mandando un token dentro de los cookies, así que solo funciona con ambientes que soporten cookies.

La característica User Web Applications es para que el servidor pueda mapear directorios de usuarios especificando el nombre del Host, un tilde y el nombre de usuario del que se desea tener acceso.

Para la utilización de ésta característica se debe de tomar en cuenta que:

1. Cada aplicación web para usuarios, será desplegada con las características establecidas por cualquier DefaultContext configurado para el Host.
2. No es permitido declarar más de una instancia Listener en este elemento.
3. El nombre de usuario del sistema operativo en el cual Catalina es ejecutado, debe de tener permisos de acceso a cada directorio de aplicaciones de usuarios y a sus contenidos.

Así con estas directivas es como se concluye la descripción del archivo de configuración de Tomcat. En la realización de la aplicación se busca configurar un conector para que colaboren el servidor Tomcat y Apache, para un mejor manejo de los recursos y un manejo de peticiones de alto rendimiento, ya que Apache recibe las peticiones HTML y Tomcat recibe las peticiones JSP.

5.4 Despliegue de aplicaciones.

El despliegue de aplicaciones, es la forma en que se ponen a funcionar las páginas dinámicas para que sean accedidas por los diferentes usuarios que así lo requieran, teniendo las clases ya compiladas por parte del servidor y no recompilarlas cada vez que el servidor reinicie o que reciba una petición de dichas páginas.

Para el despliegue de aplicaciones en el servidor, se requiere tener una estructura bien definida del directorio que contiene los archivos del sitio incluidos los archivos que utilizan las clases y los directorios que las contienen también definirla en el archivo de configuración del servidor Tomcat.

Jakarta Tomcat define en este directorio base el nombre de la aplicación, que será el directorio raíz en donde se almacena, dentro del cuál debe de aparecer un directorio llamado Web-Inf, que contiene las clases compiladas para ésta aplicación y un directorio lib, que contiene los archivos JAR que va a utilizar la aplicación.

Éste directorio almacena también un archivo descriptor de despliegue, llamado web.xml que contiene un descriptor de servlets y los descriptores de mapeo para los mismos[Tomcat, 12].

Así es como se despliega y se estructura una aplicación, que lo podemos observar en la Figura 7.

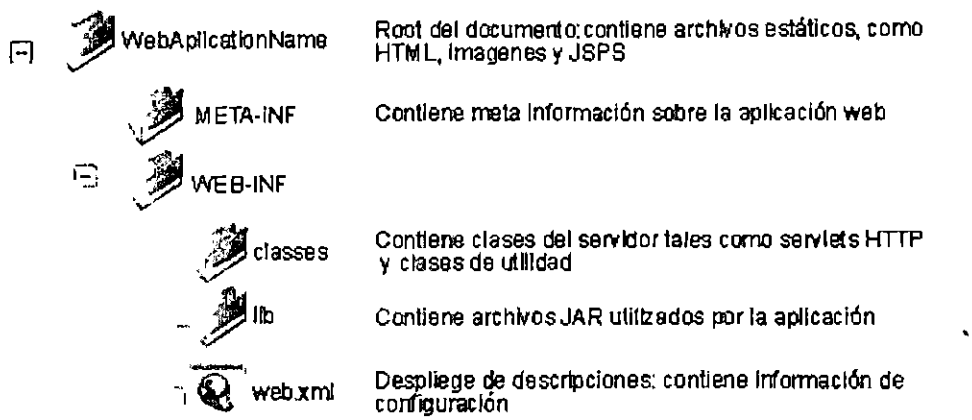

Figura 7. Estructura de directorios para el desplicgue de una aplicación

El despliegue de las aplicaciones, depende de la configuración del archivo *Web.xml*, contiene la información necesaria para definir los parámetros con los que se inicia la aplicación.

El archivo *Web.xml* define el escuchador de la aplicación, el archivo por el que se tiene acceso a la información, entre otros, que son los que definen el despliegue de la aplicación, que en la aplicación planteada, se determina con la Figura 8.

```
<?xml version=1.0 encoding= ISO-8859-1>
<!DOCTYPE web-app
    PUBLIC "..//Sun MicroSystems, Inc. // DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web app 2 3.dtd">
<web-app>
 <display-name>Congreso TTRIS</display-name>
 <description>1er Congreso Nacional de Tendencias Tecnológicas en
Redes e Ingeniería de Software</description>
 <listener>
   <listener-class>congreso.ContextListener</listener-class>
 </listener>
 <welcome-file-list>
   <welcome-file> Index.html </welcome-file>
 </welcome-file-list>
</web-app>
```

Figura 8. Descripción del archivo Web.xml

6. BASE DE DATOS MySQL

Para las empresas que manejan grandes volúmenes de información y además que ésta es modificada por varias personas al tiempo que se generan ciertos eventos, es imprescindible el uso de una base de datos que mantenga la información sin redundancias, sin riesgos y con la administración de la misma de una forma fácil y amigable.

En la actualidad, todos los sistemas de información, ya sean individuales o multiusuarios, tienen un sistema de base de datos, ya que facilita el manejo de la aplicación y la asegura por medio del mismo.

Definiciones

Una base de datos es una estructura bien organizada de datos, lo cual permite almacenar, modificar y borrar datos desde una lista del mercado, una galería o un sistema de empresa en red.

Una definición formal de una base de datos "*un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada*"[Date, 16].

Para acceder, agregar y procesar los datos dentro de la base de datos, se requiere un sistema administrador de bases de datos que es "*un sistema computarizado para guardar registros, que permite a los usuarios recuperar y actualizar esa información con base en peticiones*"[Date, 16].

Las peticiones estándar que se manejan en los sistemas manejadores de bases de datos se componen de sentencias por medio de un lenguaje SQL (Structured Query Languaje), el cual delimita lo que se va a hacer en la base de datos por medio de los comandos introducidos por el usuario.

SQL

El lenguaje SQL, se compone de consultas básicas de recuperación de datos (*Select*),

actualización de registros (*Update*), borrado de datos (*Delete*) y la adición o inserción de nuevos elementos (*Insert*).

El lenguaje SQL se ha convertido en estandar para la administración de una base de datos, ya que todas las bases de datos soportan tal lenguaje y así se pueden realizar las diferentes acciones en la base de datos.

Las sentencias SQL se estructuran por medio de palabras reservadas, como *Select, Insert, Update* y *Delete*, que son un lenguaje de pseudocódigo, ya que es fácil de entender y de aprender por la lógica fácil que se establece en sus estructuras.

La forma de interactuar una base de datos con un sistema administrador de la misma, es de la forma que se muestra en la Figura 9.

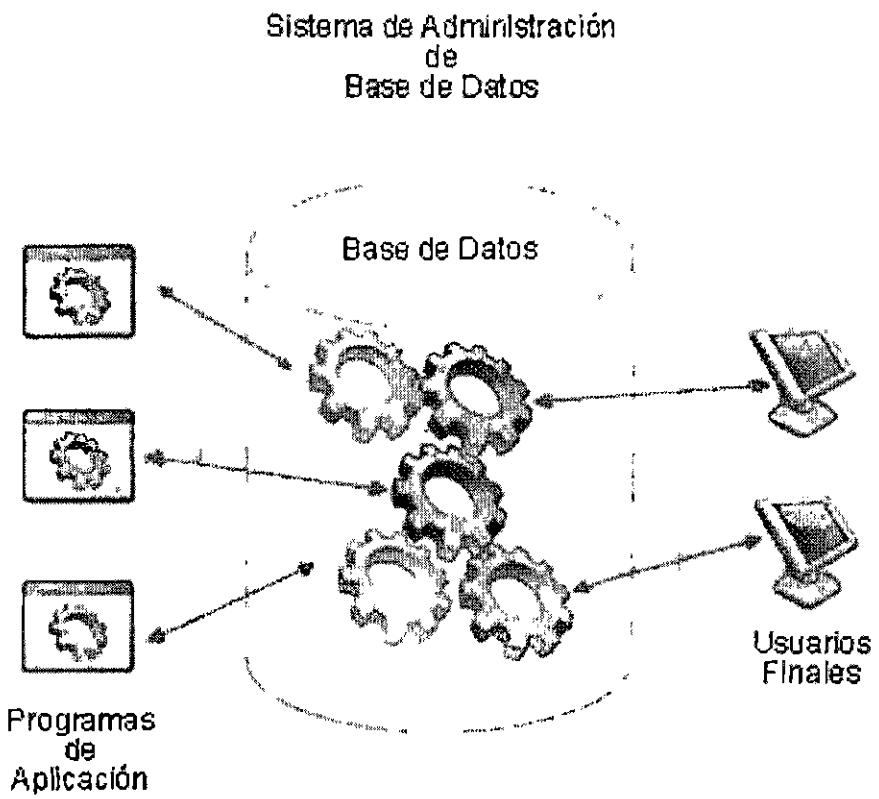

Figura 9. Imagen simplificada de un sistema de base de datos. [Date, 16]

Existen diferentes administradores de bases de datos como Informix, Oracle, access, SQL Server, MySQL, el cual se utilizó para esta aplicación por ser software libre y que viene distribuido en Linux Redhat.

MySQL, surge de la corporación MySQL AB, la cual proporciona dos tipos de licencia, la primera distribución es open source, la cual es utilizada bajo la licencia de GNU y la segunda, ofrece por una cantidad de dinero la licencia con ciertos beneficios para el cliente que la adquiera. Es una base de datos relacional, almacena la información en tablas separadas, es confiable, rápida y está en mejora continua con la cercanía de los usuarios y colaboradores.

La base de datos trabaja en un ambiente cliente servidor, ya que consiste en un sistema multi-hilo que opera en un servidor SQL, que soporta varias librerías, diferentes Back-ends y una amplia gama de interfaces.

6.1 Características y ventajas

Entre algunas de sus características y ventajas se destacan:

- Portabilidad, ya que está escrita en C y C++, probada con diferentes compiladores, trabaja con diferentes plataformas y está diseñado para trabajar en un ambiente de red con base a un modelo cliente/servidor.
- Tipos de datos compatibles, cuenta con tipos de datos enteros de 1, 2, 3, 4 y tipos long de 8 bytes, más los tipos de datos flotantes, dobles, carácter, varchar entre otros.
- Funciones y sentencias ya que es compatible con la sentencia SELECT y opera con el lenguaje de consultas SQL.
- Seguro, ya que proporciona una contraseña de privilegios, la cual permite un sistema de autentificación en cualquier aplicación que se desarrolle.
- Conectividad, ya que un cliente se puede conectar a través del protocolo TCP/IP en una plataforma de sockets. Por otro lado se permite la conexión con base a un

puente o conector ODBC, en la aplicación de éste trabajo, la conexión se establece por medio de un conector.

- Los datos pueden compartirse, ya que puede estar en alguna aplicación distribuida y además se pueden crear nuevas aplicaciones que tengan el acceso a la misma base de datos.
- Reduce redundancia, ya que los datos que se almacenan, pueden ser controlados e integrados en un solo sistema de base de datos y no guardar la misma información en distintos lugares.
- Mantiene Integridad, que son las reglas de negocio para que se guarde correctamente los datos necesarios por la aplicación y no datos incorrectos.
- Cumple la seguridad, ya que establece reglas de seguridad para las transacciones que se realizan y èrmite el rollback de las mismas que no se realizan completamente por cualquier factor externo a la base de datos.

6.3 Administración

La administración de la base de datos en lo que respecta a esta aplicación, consiste en la instalación y configuración de la base de datos. Linux tiene la base de datos ya incluido con su programa de instalación, lo cual evita el proceso de instalación de la base de datos, ya que se instala en la imagen de la instalación del sistema operativo.

El comando *mysqladmin* es relativo a la administración de la base de datos, con lo cual el usuario root, el administrador, crea la base de datos en las que se determina la ubicación, los usuarios que tendrán acceso a las bases de datos y los permisos que tendrán cada uno de ellos.

La Figura 10 muestra éste comando ejecutándose en el servidor de la aplicación.

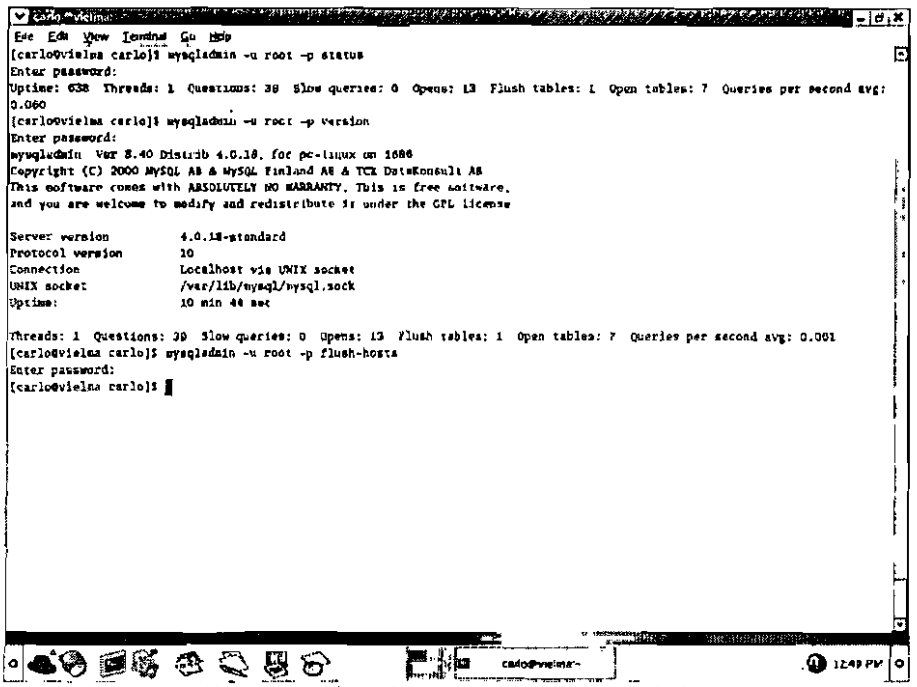

Figura 10. Ejecución del comando mysqladmin.

El comando *mysql* da el acceso a la base de datos, dependiendo de los parámetros a los que se hace referencia para establecer la conexión, se determinan los permisos de los comandos que serán permitidas para la base de datos.

En la Figura 11 se muestra la ejecución de algunas consultas ejecutadas en el sistema administrador de base de datos del sistema operativo.

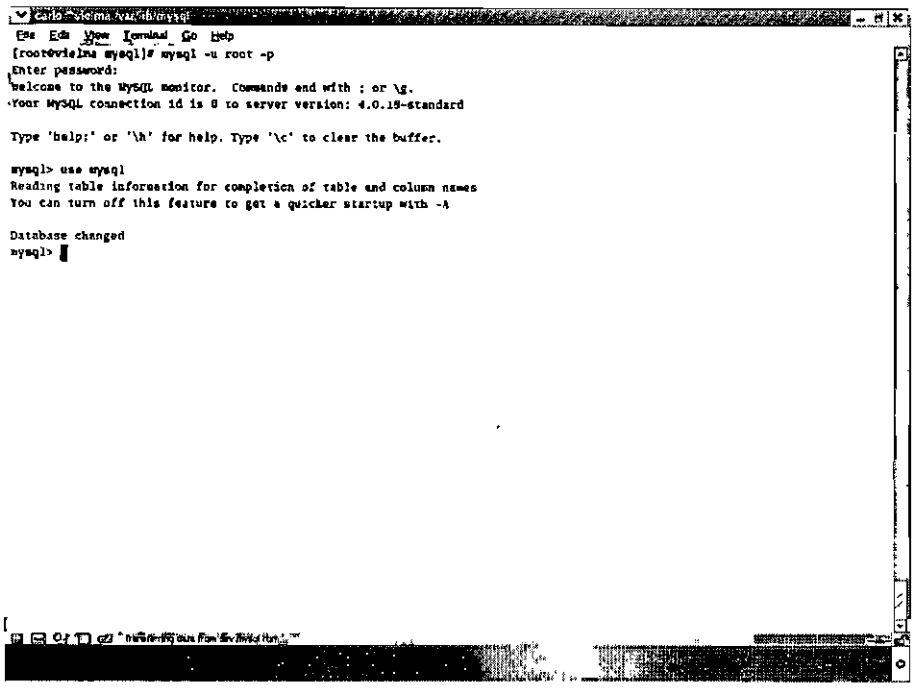

Figura 11. Ejecución de consultas con mysql.

6.4 *Seguridad*

La seguridad en la base de datos es garantizada para todos los usuarios al inicio, ya que crea dos usuarios root, uno para el host local del servidor pero con nombre de dominio *localhost* y la segunda pone como dominio la dirección IP, pero no tienen ninguna protección de contraseña, así que cualquier usuario que desee ingresar a la base de datos puede hacerlo.

Para establecer la contraseña para root se utiliza el comando de administración, *mysqladmin* para asignar el password y se asigna a las dos cuentas establecidas.

Se recomienda que se le den derechos de actualización en la tabla de usuarios a un usuario administrador que no sea el usuario root, ya que se podrían alterar los valores preestablecidos para la tabla de usuarios y se ejecuten acciones que no se pretenden.

En caso de hacer una consulta indebida como borrar usuarios de la tabla de administración, se debe restaurar o actualizar la versión del sistema administrador de la base de datos.

En la Figura 12 se muestra la estructura de la base de datos de usuarios con sus contraseñas.

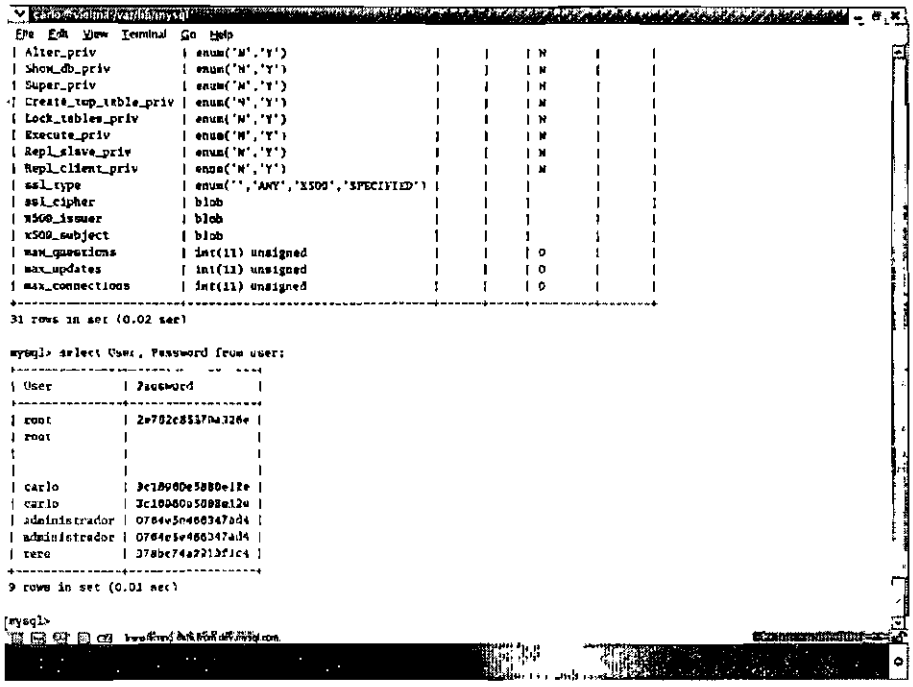

Figura 12. Usuarios con contraseña dados de alta en la base de datos.

En la aplicación planteada se genera una base de datos, en las Figuras 13 se muestran las bases de datos del sistema

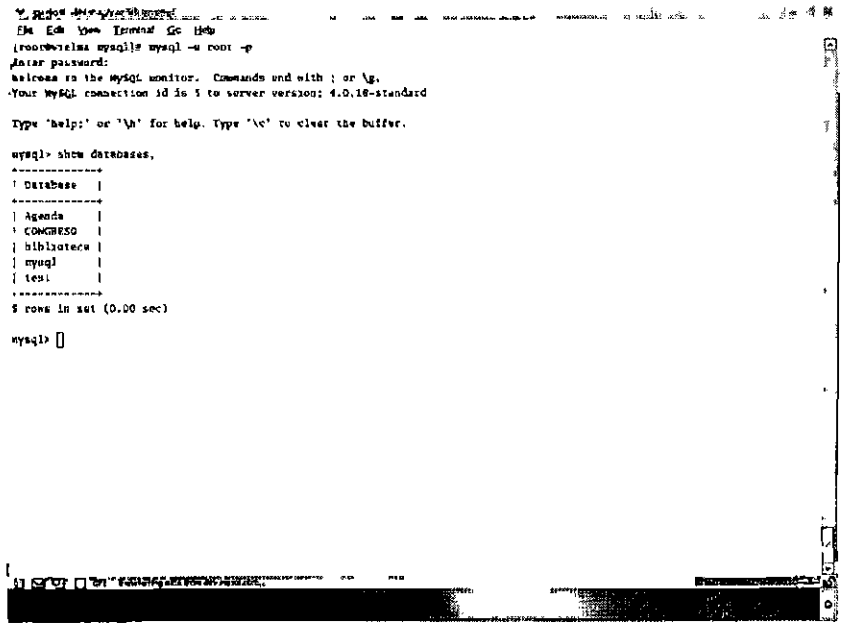

Figura 13. Bases de datos del sistema.

7. DESARROLLO DEL SISTEMA

Con base a lo descrito en la teoría y al objetivo de esta tesis, se debe presentar un modelo aplicable en el desarrollo de un sistema de información en Internet que resuelva la necesidad de divulgar y procesar información necesaria para la realización de un congreso. Se plantea una Aplicación Web que atiende la problemática de difusión, inscripciones y administración del sitio Web, tomando como referencia otros sitios de Internet que manejan éste tipo de procesos para la inscripción y verificación de usuarios.

La aplicación propuesta en este trabajo, consiste en una aplicación dinámica que soporta la inscripción de participantes (asistentes) y ponentes a un congreso, interactuando con una base de datos, la página es de información y divulgación del mismo, así como se maneja una sección de administración para la confirmación de las inscripciones.

La creación del sitio se diseñó con base al modelo propuesto para el desarrollo de aplicaciones Web de JAVA, el Modelo – Vista – Controlador (MVC), el cual establece que la aplicación se divide en tres secciones, la primera que contienen la lógica de negocio, desarrollada por programadores expertos en el lenguaje JAVA; la segunda que es la vista, que es la imagen del sitio, la estructura del mismo, desarrollada por expertos en diseño, lo cual no afecta en el desarrollo del sistema. El control de la aplicación, la tercera sección, es la interacción que hay entre el modelo y las vistas, la respuesta generada con base a los eventos que surjan.

A continuación, se describe el desarrollo del sitio Web por medio del análisis al modelo propuesto.

7.1 Análisis de Requerimientos

Los requerimientos del sistema son la parte medular para el desarrollo de una aplicación que cumpla con las expectativas y necesidades del usuario. Modificado

Para la creación del sistema de difusión e inscripción en Internet para un Congreso, los requerimientos del usuario son los siguientes:

1. Una aplicación Web para dar difusión al primer Congreso en tendencias Tecnológicas en Ingeniería de Software y Redes.
2. Permita la inscripción de alumnos de la facultad o de otras instituciones
3. Permita la inscripción de ponentes, al mismo tiempo que puedan enviar su documento.
4. El administrador del sistema podrá modificar el estatus del alumno o ponente cuando éste haya echo su pago.
5. Debe contemplar una interfaz con menús, del cual se podrá acceder a información de costos del congreso, requisitos, etc. Así como acceder a inscripciones y administración del sistema.

Requerimientos del Sistema: por ser una aplicación web se requiere de:
Un servidor Web httpd
Sistema Operativo Linux para mantener un mejor control de acceso.
Se requiere de una interfaz que soporte gráficos y animaciones en web.
Sistema Manejador de Base de Datos
Un servidor para páginas dinámicas

Dentro de la aplicación, se establecen los requerimientos por medio de un menú general, del cual se derivan los eventos o requerimientos pertinentes a los procesos a desarrollar dentro de la aplicación.

A continuación se muestra en la Tabla 7.0 los diferentes requerimientos y procesos a realizar en la aplicación.

Tabla 7.0 Descripción de requerimientos generales

PROCESO	TIPO	DESCRIPCIÓN
Información general	Liga	Es un link a la página de información del sistema, se despliega una página principal y es un ancla al inicio de la misma.
Costos	Liga	Es un ancla a la página de Información del congreso.
Inscripciones	Liga	Es un Link que nos referencia a una página nueva que es en donde se realizarán las Inscripciones a Ponentes y Participantes.
Ponencias	Liga	Es un Link a la página de información sobre los procedimientos para la escritura de las ponencias.
Programa	Liga	Es un ancla al programa que se seguirá para el desarrollo del evento.
Administración	Liga	Es un Link a una página que requiere autorización para ser accedida y se pretende la edición y borrado de participantes y ponentes.

En el requerimiento de inscripciones, se generan otros dos procesos, los cuales consisten en la inscripción de participantes y ponentes para el congreso, estos requerimientos se describen en la Tabla 7.1.

Tabla 7.1 Descripción requerimientos de inscripción

PROCESO	TIPO	DESCRIPCIÓN
Part/Pon	Opción Radio	Esta opción establece que proceso es el que se realizará, si se va a inscribir un participante o un ponente.
Participante	Proceso	Se despliega la forma, se introducen los datos y se insertan en la tabla correspondiente.
Ponente	Proceso	Se despliega la forma, se introducen los datos y se insertan en la tabla correspondiente.

Otro de los requerimientos que generan procesos, es el de administración, el cual genera los requerimientos de autenticación, para acceder a esta parte del sitio y después permite las opciones de edición o eliminación de preinscritos, éstos se describen en la Tabla 7.2.

Tabla 7.2 Descripción requerimientos de administración

PROCESO	TIPO	DESCRIPCIÓN
Autenticación	Proceso	Muestra la forma, se introducen clave y contraseña y los compara con el resultado de una consulta, si son iguales, se despliega la nueva forma.
Editar	Proceso	Se despliega la forma, obtienen los datos y se actualiza en la tabla correspondiente.
Borrar	Proceso	Se despliega la forma y se borra en la tabla correspondiente.

Así se concluyen los procesos que se realizarán con el sistema, para el despliegue de información, la inscripción de participantes y ponentes al congreso y la confirmación o eliminación del administrador.

Los procesos de HTTP, se programaron con HTML, para crear páginas Web, mientras que las animaciones se manejaron con Flash, las cuales controlan sus eventos por medio de Action Script, los eventos para determinar despliegues de participantes y ponentes se realizaron por medio de JavaScript.

A continuación se muestra cómo se plantea la Base de Datos para la aplicación, en la cual se almacenan los datos requeridos por el sistema para obtener la información de los diferentes visitantes y del administrador.

7.2 Descripción de la base de datos

La base de datos consta de dos tablas, la primera almacena los datos y propiedades de los inscritos, en donde el visitante se da de alta por medio de una página Web y el administrador será el único con acceso a los campos de la misma para confirmar al inscrito.

A continuación se muestra en la Tabla 7.3 los detalles de la tabla de inscritos en la base de datos.

Tabla 7.3 Detalles de la tabla inscritos

Campo	Descripción	Tipo
Apaterno	Campo para almacenar el apellido paterno del asistente.	Varchar
Amaterno	Campo para almacenar el apellido materno del asistente.	Varchar
Nombres	Campo para almacenar los nombres del asistente.	Varchar
Institución	Campo para almacenar la procedencia de la institución educativa del asistente.	Varchar
Email	Campo para almacenar la dirección electrónica del asistente.	Varchar
Participante	Campo que indica si el asistente es participante (1/0).	Int
Ponente	Campo que indica si el asistente es ponente (1/0).	Int
Ponencia	Campo que almacena el nombre de la ponencia del asistente en caso de que sea ponente.	Varchar
Inscrito	Campo que almacena un valor para determinar si el asistente ya está o no inscrito, este campo solo es editable por el administrador.	Int

La segunda tabla es para tener un control de acceso a la administración del sitio, esta tabla sólo contiene dos columnas de las cuales una es el usuario y la segunda es la contraseña almacenada para permitir modificar la base de datos.

La tabla administrador se describe en la Tabla 7.4.

Tabla 7.4 Detalles de la tabla administrador

Campo	Descripción	Tipo
Usuario	Campo que define el usuario administrador.	Varchar
Contraseña	Campo que contiene la contraseña para acceder al sistema como administrador	Varchar

7.3 Diseño del Sistema

En cuanto al diseño del sistema, se retoman los requerimientos, transportándolos a procesos relacionados con la codificación y el lenguaje de programación a utilizar y cómo se despliegan cada uno de ellos, describiendo sus procesos como tipo de elementos respectivos.

Se toma como base al modelo propuesto por JAVA MVC, en primer lugar, se describe la Vista de la aplicación, en la cual se destacan los componentes del diseño del sitio Web y los eventos del Control que proporcionan la comunicación con el Modelo.

La descripción del modelo se aprecia en la Figura 14.

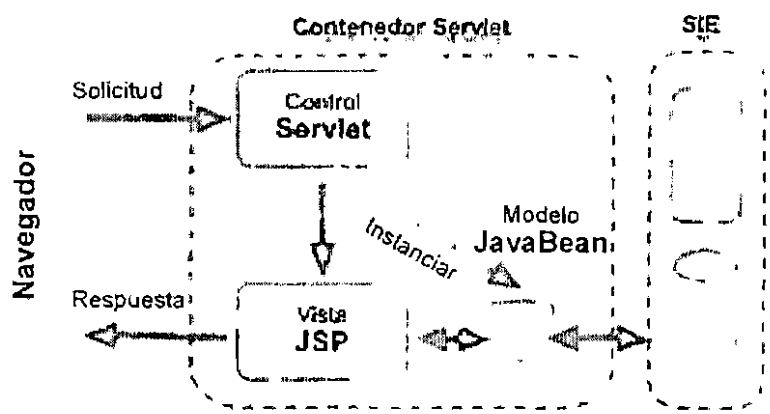

Figura 14 Descripción MVC.

Se muestra en la Figura 15 la representación del modelo con UML

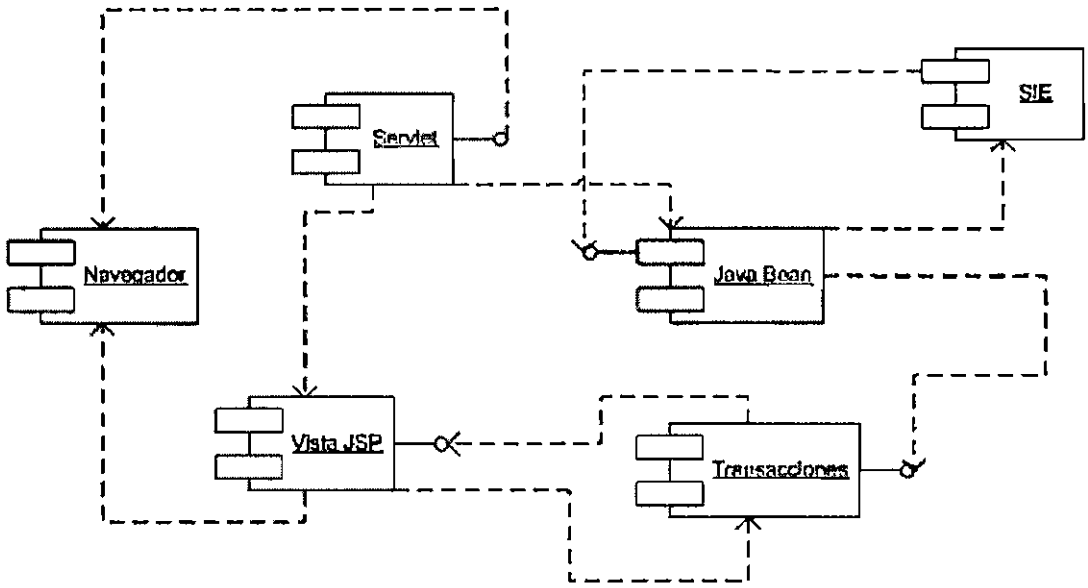

Figura 15. Descripción MVC.

7.3.1 Vista y Control

Para los procesos planteados en los requerimientos, se toma en cuenta la presentación que se tendrá para cada uno de ellos, los cuales se describen según la vista o presentación que tendrá se tendrá para los usuarios, que consiste en un menú y su contenido se describe a continuación.

En la Tabla 7.5 se describen los procesos de diseño del sistema con base a los requerimientos.

Tabla 7.5 Descripción del diseño de los requerimientos		
PROCESO	**TIPO**	**DESCRIPCIÓN**
Información general	Link	Se realiza por medio de un menú de Flash utilizando ActionScript, utilizando Anclas a la parte del documento escrito en HTML correspondiente.
Costos	Link	Se realiza por medio de un menú de Flash utilizando ActionScript, utilizando Anclas a la parte del documento escrito en HTML correspondiente.
Inscripciones	Link	Se realiza por medio de un menú de Flash utilizando ActionScript, utilizando Anclas a un doeumento nuevo que se despliega mostrando dos frames que tendrá uno la función de menú y otro de zona de despliegue.
Ponencias	Link	Se realiza por medio de un menú de Flash utilizando ActionScript, utilizando Anclas a un documento nuevo escrito en HTML correspondiente.
Programa	Link	Se realiza por medio de un menú de Flash utilizando ActionScript, utilizando Anclas a la parte del doeumento escrito en HTML correspondiente.
Administración	Link	Se realiza por medio de un menú de Flash utilizando ActionScript, utilizando Anclas a un documento nuevo que se despliega mostrando una página para la autentieación de usuario y mandar el respectivo mensaje o autorización.

Para la vista se toma en cuenta un diseño con base a páginas HTML, los cuales se distribuyen en la presentación, para mostrar una sección del sitio, atendiendo a los eventos que se requieran.

Para el proceso de inscripciones la vista consiste en dos frames que contienen los formularios para obtener los datos de los usuarios, para esto se requiere una nueva página de opciones y procesos para su realización, en la Tabla 7.6 se describen los procesos para las opciones de inscripción.

Tabla 7.6 Descripción de la vista de inscripción		
PROCESO	**TIPO**	**DESCRIPCIÓN**
Part/Pon	Opción Radio	El frame superior que sirve de menú principal para escoger una opción, que con el manejo de eventos se realiza el despliegue correspondiente en frame inferior.
Participante	Form	Forma para recuperar información acerca del asistente al congreso.
Ponente	Form	Forma para recuperar información acerca del asistente al congreso.

En cuanto a la presentación del administrador lo primero que se pide es la autenticación del mismo por medio de una Forma, que será su vista y enseguida el despliegue de la consulta de la base de datos establecida por una página JSP y finalmente atender las opciones del administrador, lo cual se muestra en la Tabla 7.7.

Tabla 7.7 Descripción del control de la administración		
PROCESO	**TIPO**	**DESCRIPCIÓN**
Autenticación	Proceso	Muestra la forma, obtiene los datos y los compara con el resultado de una consulta y si son iguales, se despliega la nueva forma.
Editar	Proceso	Se despliega una tabla con información del asistente y una imagen botón para la detección de eventos Onclik para editar al asistente.
Borrar	Proceso	Se despliega una tabla con información del asistente y una imagen botón para la detección de eventos Onclik para borrar al asistente.

Ya descritos los requerimientos del sistema y descritos en el diseño la Vista y el Control de la aplicación, se construyó el sitio, en cuanto al Modelo, que es el que recibe los datos del servlet generado e instancia las clases necesarias para llevar la secuencia del programa de una forma estructurada, se describe su procedimiento a continuación.

El modelo se plantea en la lógica del negocio y los programas a realizar, para lo que se utilizan JSP's, servlets, Beans y una clase externa que interactúa con la base de datos del sistema. Esto se realizó de la forma en que se describe en la construcción del modelo de la aplicación.

7.3.2 Modelo y control

El modelo de la aplicación consiste en la creación de la parte funcional del negocio interactuando con la vista por medio del control, el modelo es la clase en lenguaje nativo JAVA para la creación de peticiones a la base de datos y obtener la información que se genere de dichos procesos, que regresan a la vista como páginas que serán mostradas en el navegador, una de las herramientas para implementar el control son las páginas JSP.

Los eventos que son la parte del control de la aplicación se generan de diversas formas, éstos son de entrada, las peticiones de los links que utiliza el usuario, que se programan en HTML y animaciones en Flash y proporcionan eventos en ActionScript y JavaScript.
La lógica del programa o el Modelo se realiza por medio de JSP, que es código JAVA incrustado en páginas HTML, que serán atendidas por un servidor Tomcat; las páginas JSP, interactuarán con un componente Bean que será el medio de comunicación que retendrá la información del asistente para ser ingresado o actualizado en la base de datos. Éste interactúa con una clase para el manejo de la base de datos, para obtener la conexión y liberarla, así como para la manipulación de la misma.

La Figura 16 y 17 muestra la división del MVC separándolo en la vista (Parte frontal) y el modelo (Parte trasera).

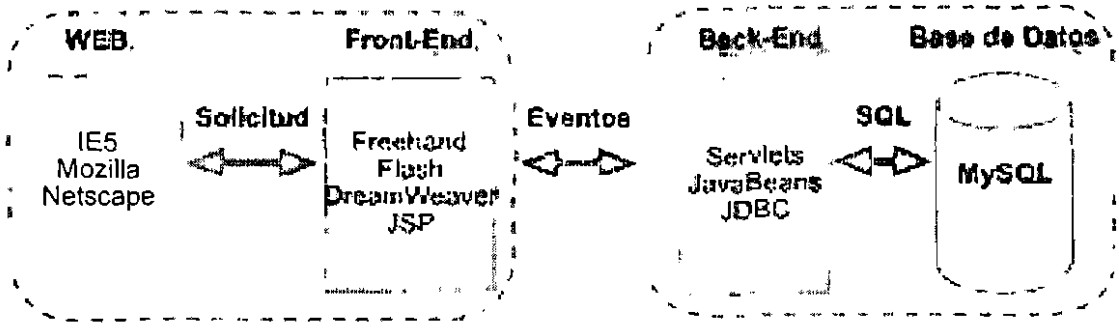

Figura 16. El Back-end y Front-End en la estructura MVC.

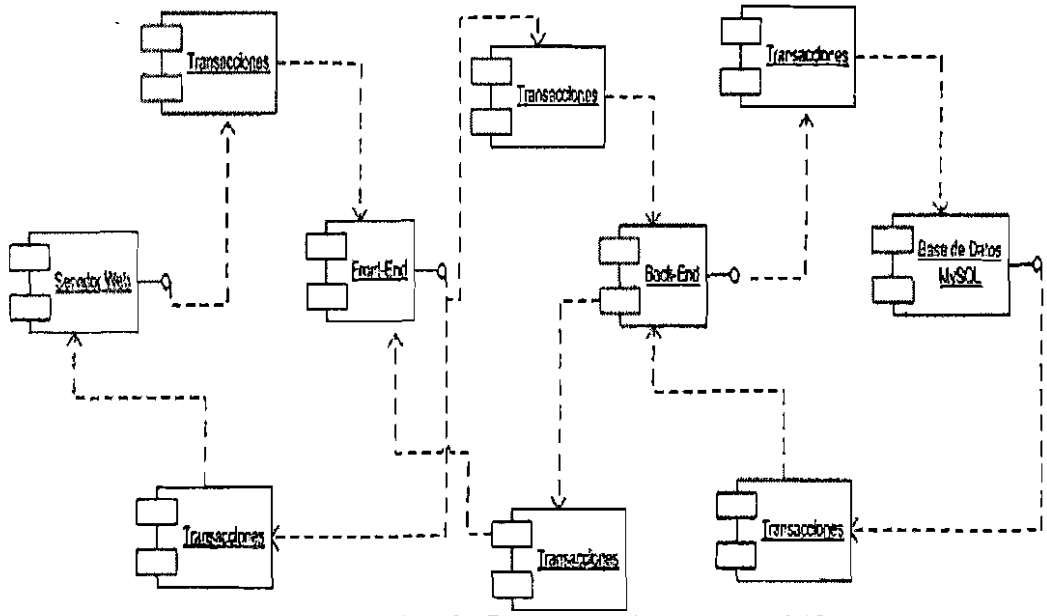

Figura 17. El Back-end y Front-End en la estructura MVC.

7.3.3 Implementación del sistema

La implementación del sistema para la divulgación de un congreso, así como la inscripción de participantes al mismo, surge de una necesidad en la Facultad de Informática para la realización del Primer Congreso Nacional de Tendencias Tecnológicas en Redes e Ingeniería de Software, que es en donde se implementa la teoría planteada.

En la primera pantalla, se genera la información pertinente para mostrarla en la página principal; se estructura en una tabla, la cual en la parte superior contiene la animación de cabecera que contiene en el lado izquierdo el logotipo de la UAQ, en el centro el título del congreso y en la parte derecha el logotipo de la Facultad de Informática.

El siguiente renglón contiene dos columnas de datos, en la primera del lado derecho, se tiene el menú que es otra animación FLASH, que servirá como menú principal de navegación. La siguiente columna presenta el contenido de la información del congreso.

Esto se observa en la Figura 18.

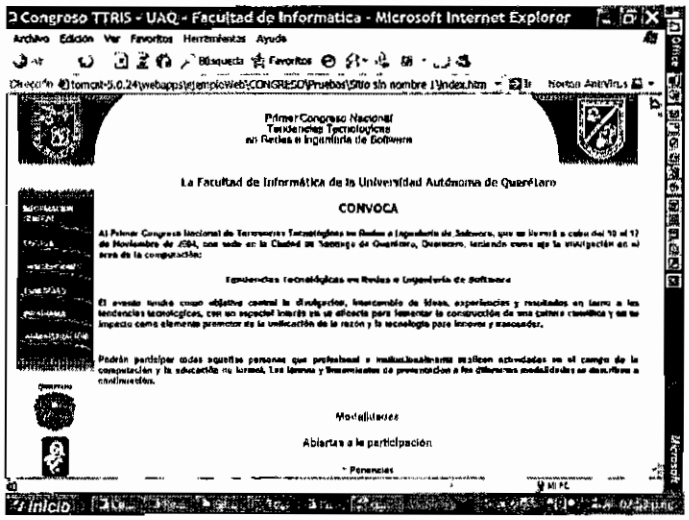

Figura 18. Página principal de la aplicación

Para la interfaz de usuario en la vista de inscripción se presentan dos frames como lo muestra la Figura 19.

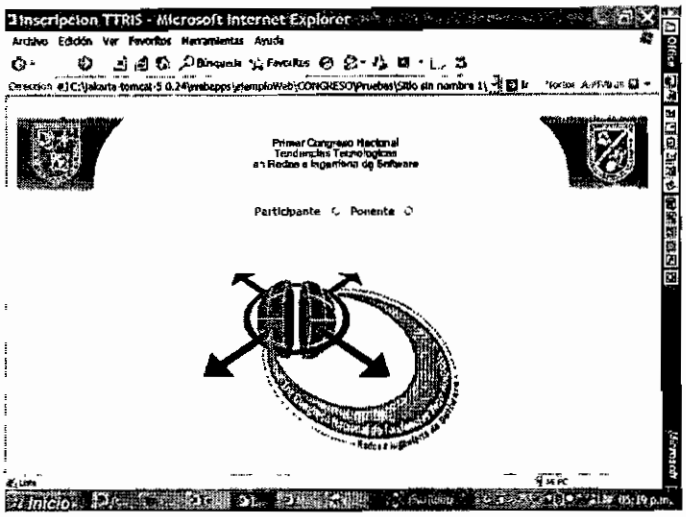

Figura 19. Presentación de Inscripciones.

Para cada usuario se establece la interfaz de inscripción como se muestra en las figuras siguientes; la Figura 20 es para la inscripción de participantes y la Figura 21 para la inscripción de ponentes.

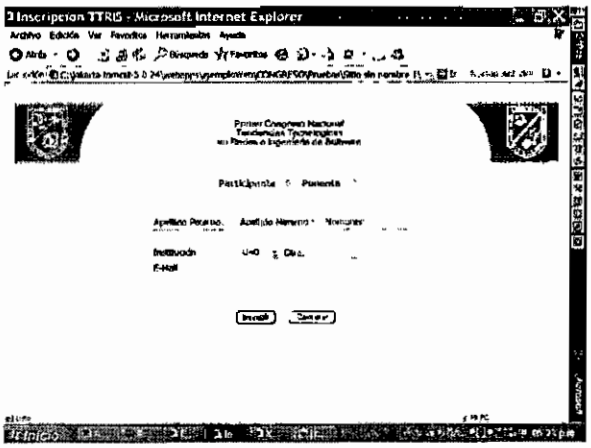

Figura 20. Inscripción participante.

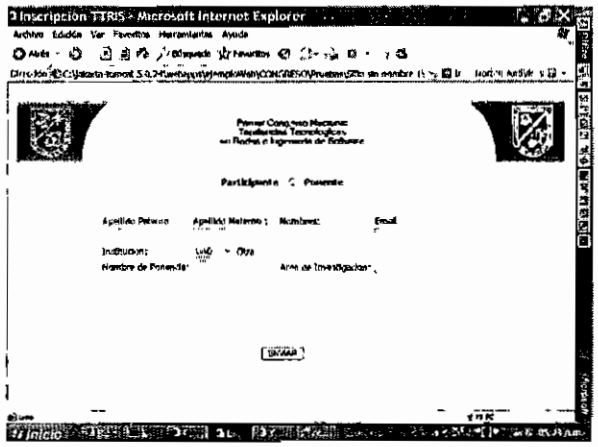

Figura 21. Inscripción ponente.

En la Figura 22 se muestra el código JSP que permite guardar la información capturada por el usuario en la base de datos, se señalan las operaciones con el bean, con la interfaz (html) y los scriptlets.

```
<%@ page language="java" import="java.util.*,
java.sql.*,CongresoTTRIS.BeanInscripcion.*" %>
<html>
    <head>
        <title>Confirmacion de la Insercion de un Participante
        </title>
    </head>
    <body>
        <h1><center>Participante</center></h1>
        <table align="center" cellpadding="2" cellspacing="2"
        border="0" width="50%" bgcolor="#999999">
         <tr>
            <th width="38%" align="justify">NOMBRE:</th>
                <td width="62%" align="justify"><%=
            request.getParameter ("nombres") %></td>
         </tr>
         <tr>
            <th align="justify">APELLIDO PATERNO:</th>
                <td align="justify"><%= request.getParameter
                ("apaterno")%></td>
         </tr>
         <tr>
           <th align="justify">APELLIDO MATERNO:</th>
               <td align="justify"><%= request.getParameter
               ("amaterno") %></td>
         </tr>
         <tr>
            <th align="justify">INSTTITUCION:</th>
               <td align="justify"><%= request.getParameter
               ("institucion") %></td>
         </tr>
         <tr>
            <th align="justify">EMAIL:</th>
```

Inclusión de librerías

Scriplet Recepción de datos

```
                <td align="justify"><%= request.getParameter
                ("email") %></td>
            </tr>
        </table>
<%
    CongresoTTRIS.BeanInscripcion.BaseDatosTTRIS bdt=new
    CongresoTTRIS.BeanInscripcion.BaseDatosTTRIS();
    CongresoTTRIS.BeanInscripcion.BeanInscripcion bi=new
    CongresoTTRIS.BeanInscripcion.BeanInscripcion();
try{
        bi.setNombre(request.getParameter ("nombres"));
        bi.setApaterno(request.getParameter ("apaterno"));
        bi.setAmaterno(request.getParameter ("amaterno"));
        bi.setInstitucion(request.getParameter ("institucion"));
        bi.setEmail(request.getParameter ("email"));
        bi.setParticipante(1);
        bi.setPonente(0);
        bi.setPonencia(" ");
        bdt.init();
        bdt.inserta(bi);
        bdt.end();
}catch(Exception e){
        bdt.end();
        out.println("<center>Debe de llenar todos los campos y que
        conincidan con el tipo de dato<br><br></center>");
        out.println("El error es: "+e);}
%>
 <hr>
 <center>
  <b><a href="index.htm">HOME</a></b>
</center>
</body>
</html>
```

Scriplet Instancia Manciador BD

Scriplet Instancia Bean

Guarda datos En el Bean

Guarda el bean en la Base de Datos

Figura 22. Descripción de Interfaz.

La parte del Administrador establece una autentificación como se muestra en la Figura 23.

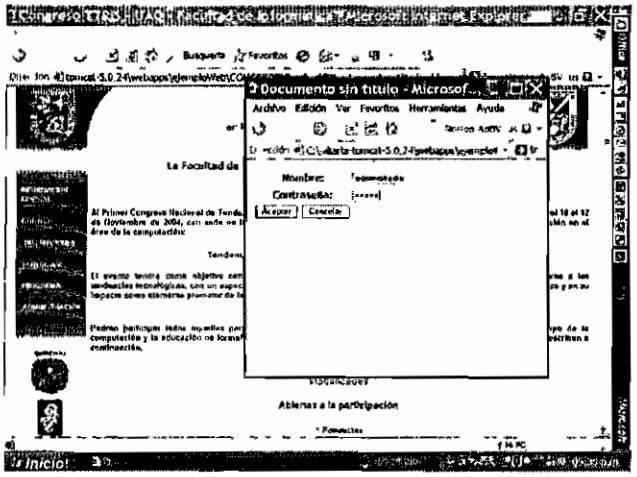

Figura 23. Autentificación de la Administración.

En la administración, se despliega la información de la base de datos, dando las opciones que se pueden realizar en la aplicación, esto se muestra en la Figura 24.

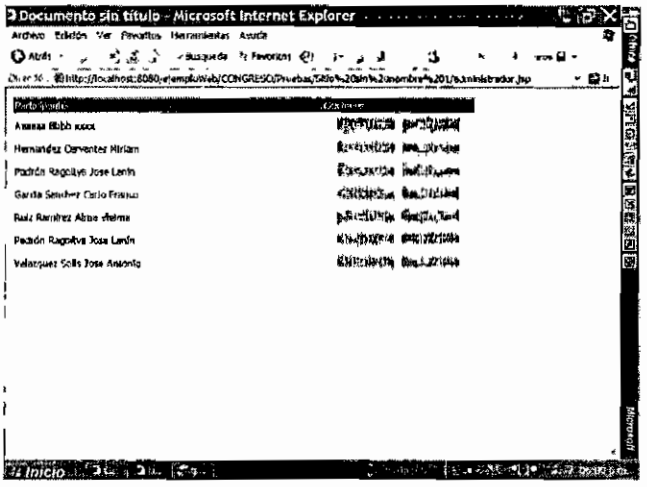

Figura 24. Sección de Administración.

Así se establecen los asistentes y se presentan los registros en la base de datos, por lo cual se puede decir que ya está funcionando la aplicación en la que se insertan los datos en las tablas de la base de datos.

8. CONCLUSIÓN Y TRABAJOS FUTUROS

En el desarrollo de aplicaciones Web es importante considerar los siguientes elementos, modelo de desarrollo, plataforma a utilizar, servidor y software de desarrollo, para lograr una gestión segura de la información.

La tecnología Java en conjunto con una plataforma como Linux, el servidor web apache y el servidor Tomcat para páginas dinámicas java, permite corroborar la hipótesis planteada para este trabajo en relación a la seguridad de las aplicaciones web dinámicas. Lo anterior se logró al aplicar el modelo MVC y las características de cada una de las herramientas y plataformas utilizadas para el desarrollo de la aplicación.

El manejo de un modelo como lo es MVC, garantiza la separación de la lógica de aplicación y la lógica de presentación. Otro punto fundamental es que el modelo que propone JAVA (MVC), junto con el modelo de aplicación que propone la estructura de Servlet 2.2, que se puede consultar en la página de java.sun.com, se adecua perfectamente a la división de la lógica de programa con la imagen, permitiendo así dividir la aplicación en directorios lógicos, que contengan las clases del lenguaje de programación nativo y los directorios de elementos interactivos y de imagen, todo constituido en un directorio raíz.

El desarrollo de la vista, implementado por profesionistas en el área de tratamiento de imágenes, diseño gráfico, mércadotecnia y software para el diseño de páginas Web. En cambio para la lógica de aplicación, se debe desarrollar por especialistas en lenguajes de programación para el desarrollo de páginas Web, administración de bases de datos y sistemas operativos.

La vista de la aplicación debe facilitar la integración de elementos multimedia, es por eso que se utilizó el software de Macromedia (FreeHand, FireWorks, DreamWeaver, Flash), que proporcionan imágenes y animaciones que dan una interfaz de usuario amigable y atractiva.

En cuanto al modelo de la aplicación recibe los datos y les da un seguimiento a través del control, que es un JavaBean externo a la aplicación, que contiene los datos y es el parámetro a otra clase externa a la aplicación que ayudará a insertar, actualizar o eliminar elementos de una base de datos, permitiendo así la separación de la aplicación que es Web, del lenguaje de programación que interactúa con la base de datos (JAVA).

Todo lo anterior realizado por medio del control, que en la aplicación surge de los JSP compilados, que generan un servlet que lleva la secuencia de la página conforme a los eventos que el usuario transmita.

En cuanto a la transmisión de información, el MVC genera un ambiente óptimo para determinar que la información solo viaja a través de la red por medio de formularios y que el tratamiento de la misma, en base a éste modelo, es por medio del lenguaje de programación puro y externo a la aplicación, que se comunican por medio de interfaces y componentes.

Ahora, por medio del servidor y el sistema operativo que se esté utilizando, se da una seguridad de acceso a los directorios de la aplicación y al sistema operativo, haciendo la aplicación confiable de cualquier intento de acceso ilegal ya que Linux brinda un alto grado de seguridad y facilidad para la administración de los usuarios. Con respecto al servidor Web, es necesario destacar que su manejo de seguridad para usuarios es similar al de Linux, lo que lo hace un servidor confiable y fácil de configurar, incluyendo la funcionalidad para JSP y PHP, que se configuran con sus respectivos módulos.

JSP soporta el desarrollo de páginas dinámicas, lo que permite la actualización de las mismas sin tener que desmontar la página para modificarla y cómo está basado en el lenguaje de programación de JAVA y es montada del lado del sevidor, sólo se necesita que la máquina cliente tenga instalada una versión de navegador de Internet.

La programación de JSP, permite llevar un control adecuado de las aplicaciones Web, ya que nos dan un control seguro de la información generándola y requiriéndola sólo cuando

es necesario y nos da un ambiente de independencia para la generación de páginas Web, en donde sólo se requiere un programador cuando se requiere añadir alguna funcionalidad al sitio.

Trabajos Futuros

Dentro de lo que se contempla para el desarrollo de trabajos futuros derivados de éste trabajo, es transportar el modelo propuesto para la creación de un SIE (Sistema de Información Empresarial), en donde se pueda explotar toda la tecnología que nos proporcionan los sistemas de información montados en Internet.

Se contempla la realización de diferentes artículos de divulgación científica en base al desarrollo de aplicaciones que interactúen con JSP's, Servlets, JavaBeans y páginas HTML, montadas en diferentes servidores para la creación de una sola aplicación.

9. ANEXOS

Anexo A

Instalación de Linux Red Hat 8.

Para la instalación del Sistema Operativo Linux Red Hat 8, es necesario tener en cuenta cuáles son los requisitos que debe de tener la PC, en primera, para que el sistema se pueda instalar, y en segunda para que el sistema tenga un rendimiento óptimo, tales características son:

- Un procesador Intel Pentium o compatible.
- Al menos 32 MB en RAM, se recomienda 64MB, para ejecutar el entorno gráfico.
- Al menos 500MB en disco duro.
- Un lector de CD-ROM.

Estos elementos son de vital importancia para la instalación de Linux, ya que son los requisitos mínimos para que se instale un Sistema Operativo funcional en modo gráfico, pero por otro lado, se necesita mucho menos para una instalación de un Sistema Operativo funcional sin un modo gráfico, como lo sería una instalación en modo de consola, o mejor dicho, solo un Sistema Operativo en modo de terminales. En este trabajo se va a instalar Linux Red Hat en su versión 8, y será de un modo completo y detallado, ya que esta instalación, como lo permite Linux, da la opción de instalar cuantos paquetes deseemos y lo permita el hardware.

Para la instalación de Red Hat Linux, en una PC, es necesario en primer lugar elegir como se va a lleva a cabo ésta; por medio de CD's, por medio de algún servidor, ya sea HTTP, FTP o NFS, o del mismo disco duro de la máquina. Cualquiera de estos métodos es el adecuado para una instalación segura y fiable de Linux.

1. Inicio de la Instalación.

Ya que se cuenta con la distribución de Linux Red Hat en los cinco CD's correspondientes, se inicia la instalación insertando el Disco número uno y reiniciando la PC. Como la computadora sólo usará un sistema operativo, no es necesario revisar detalles a fondo para las particiones del disco duro.

2. Modo de Instalación.

Ya iniciado el proceso de instalación, se pide la forma de instalar, si en modo gráfico, en modo de texto o la ayuda general, que indica en qué consiste cada uno de los métodos de instalación. Se elige la instalación en el modo gráfico, que inmediatamente manda a una pantalla para elegir el idioma en el cual se llevará a cabo la instalación.

3. Selecciou de Teclado y Mouse.

El siguiente paso es elegir un modelo de teclado, Linux lo asigna de modo predefinido, si se desea otra configuración o modelo se selecciona y se da clic en continuar o siguiente, lo cual lleva a la selección del ratón, que igualmente que con el teclado, lo asigna por default y se puede configurar según las diferentes necesidades; ya realizado esto, se continua oprimiendo siguiente.

4. Elección del Tipo de Instalación.

En Linux hay cuatro formas de instalación, una que es la de modo de Escritorio Personal, que ésta consiste en instalar lo necesario para equipos domésticos o de oficina. La otra es el modo de Estación de Trabajo, la cual consiste en instalar lo necesario para ser Escritorio Personal más aparte otras herramientas de trabajo para la administración y desarrollo del Sistema. La tercera opción, que es la de servidor, que esta opción, instala los paquetes y herramientas necesarias para que la PC en la que va a ser instalado el Sistema Operativo funcione como servidor. Y la última opción es la Personalizada, en donde la instalación le permite al instalador, elegir cuales son los paquetes que se van a instalar.

Como ya tenemos nociones del Sistema Operativo, se selecciona el modo de Instalación

Personalizada.

5. Particionamiento.

Con respecto a la determinación de las particiones, existen varios modos de particionar, uno es eliminar todas las particiones de Linux del Sistema, que ésta permitirá guardar las particiones que no sean de Linux e instalará en las particiones que liberará la instalación. La segunda es eliminar todas las particiones del sistema, así la instalación borrará todas las particiones e instalará Linux en todo el Disco Duro. La tercera forma de particionar, es Guardar todas las particiones existentes y use el espacio libre existente.

Como ya se expuso anteriormente, que el equipo de trabajo sólo tendrá Linux como Sistema Operativo, se selecciona la segunda opción.

6. Seleccionar la Partición de Arranque.

En esta sección de la instalación, es importante asignar la memoria de arranque (BOOT) y la memoria virtual que ocupará el sistema (SWAP), así como la memoria en donde se instalará el Sistema (/ ó ROOT).

Para esto, se seleccionan 100MB del disco duro y se asigna a BOOT, 200MB a la memoria SWAP y lo que resta a (/) ó ROOT.

7. Configuración de la Red.

Para configurar la red en la instalación de Linux, es necesario que la PC esté conectada a una red LAN, para que así, se puedan asignar las especificaciones para su funcionamiento, sin embargo, si la PC, se conecta a Internet por medio de una conexión telefónica y no pertenece a ninguna red LAN, no es necesario establecer parámetros de red.

Lo primero que se pide es la dirección IP, puede ser estática, que la asigne manualmente el usuario que está instalando o dinámica, que la asigne al arrancar el sistema.

El segundo parámetro a configurar, es la mascara de red, que permitirá establecer que parte

de la IP es una red y que parte es la que ubica a una PC.

El tercer parámetro que se configura es que se tiene que activarse al arrancar el sistema.

A continuación se establece la puerta de enlace, que es la que va a proporcionar una salida a Internet.

Se configura el DNS, que es el Servidor de Nombres de Dominio, que es el que resuelve las direcciones IP de la red.

8. Confignración del Fire Wall

El paso siguiente es con relación a la seguridad que se va a manejar en nuestro sistema, y a que servicios de red se van a utilizar, se dan tres niveles para escoger y se debe seleccionar uno, los niveles son:

- Alto: No permite ningún otro servicio más que el DNS y el DHCP.
- Medio: Bloquea los accesos a los puertos que se manejan dentro de la pila de protocolos TCP/IP, y rechaza el acceso a los puertos utilizados para accesos remotos y para servidores de red.
- Ningún firewall: Este nivel no bloquea ningún servicio de red y deja abiertos los puertos de servicios de red.

También se puede personalizar que servicios son los que se van a utilizar, seleccionándolos por separado de la configuración del firewall.

En éste caso, el nivel de seguridad se selecciona alto, pero se configuran y se permiten los servicios de FTP y HTTP.

9. Selección del Soporte de Idioma.

En esta parte, se le especifica al sistema, que idiomas son los que va a manejar. Para ésta instalación se configura con el idioma Inglés.

10. Selección de uso horario.

Se especifica la hora dependiendo de la región en donde se encuentre.

11. Configuración de cuentas de usuario.

En esta sección, se especifica una contraseña de root y se especifican a los usuarios que van a poder ingresar al equipo sin ser administradores o mejor dicho sin los privilegios de root.

12. Selección de paquetes.

La selección de paquetes, consiste en seleccionar y personalizar los paquetes que se van a instalar en la PC, por lo general aparecen seleccionados los paquetes orientados al tipo de instalación que se hizo anteriormente. Sin embargo se pueden seleccionar todos o incluso quitar algunos que ya estén seleccionados.

En este punto es donde inicia la instalación del sistema operativo y los diferentes paquetes, el tiempo depende de la velocidad de la computadora y de los paquetes seleccionados.

13. Creación de un disco de inicio.

Terminada la instalación del sistema operativo y de los paquetes en la computadora, el siguiente paso es crear un disco de inicio, que es muy recomendable, ya que éste le permitirá el acceso a su sistema operativo, si llega a fallar o si se pierde parte de la configuración.

14. Configuración de la tarjeta de video y monitor.

La tarjeta de video se detecta automáticamente, sólo es un paso para la confirmación por parte del usuario de que no sea otra tarjeta de video la que se instale. Y por otro lado, la configuración del monitor y de la resolución de la pantalla; en esta etapa se elige si el inicio de la PC es en modo grafico o en modo texto.

15. Fin de la instalación.

Hasta este punto ya se instaló el sistema operativo, se quitan los discos de instalación y se reinicia el equipo, lo siguiente es la configuración de la fecha y hora y la instalación de paquetes extra o actualización el sistema operativo.

Anexo B

B1. Instalación de la JVM

Para la instalación de la máquina virtual de JAVA, es necesario descargarla del sitio de sun (java.sun.com). en seguida, se siguen los procedimientos que se describen a continuación.

1.- Creamos el directorio de donde se va a localizar nuestra JVM

```
cd /usr/local
mkdir javal_4
```

2.- Copiamos el j2sdk-1_4_2_03-linux-i586.bin a /usr/local/java1_4:

```
cp  j2sdk-1_4_2_03-linux-i586.bin  /usr/local/javal_4/
```

3.- ejecutamos el archivo

```
./j2sdk-1_4_2_03-linux-i586.bin
```

4.- Se crea el directorio de java, al cual le hacemos una liga para el fácil manejo de directorios.

```
ln -s j2sdk1.4.2_03 java
```

5.- Ahora se modifica el archivo profile para agregar la variable de entorno path y agregar la variable JAVA_HOME;

```
export JAVA_HOME=/usr/local/javal_4/bin/
CLASSPATH=$CLASSPATH:$JAVA_HOME
```

y de este modo se instala la maquina virtual de java.

B2. Instalación de Tomcat

Para instalar el contenedor de páginas dinámicas que atenderá las peticiones JSP, es necesario descargar el servidor de Apache.org y se realizan las operaciones descritas a continuación:

1.- Mover el archivo jakarta-tomcat-4.1.27.tar.gz a "/usr/local".

```
mv  jakarta-tomcat-4.1.27.tar.gz   /usr/local/
```

2.- Se descomprime el archivo

```
tar -xvzf  jakarta-tomcat-4.1.27.tar.gz
```

Se crea la carpeta jakarta-tomcat-4.1.27

3.- Se genera una liga a nuestro tomcat

```
ln -s jakarta-tomcat-4.1.27 tomcat
```

4.- Se modifican las variables de entorno para agregar a TOMCAT_HOME en el .profile

```
export TOMCAT_HOME=/usr/localtomcat/
CALSSPATH=.$CLASSPATH:$JAVA_HOME:$TOMCAT_HOME
```

y tenemos el servidor de paginas dinámicas jsp en el sistema operativo.

B3. Instalación de Jakarta-Ant

1.- Mover el archivo jakarta-ant-1.5-bin.zip a /usr/local.

```
mv   jakarta-ant-1.5-bin.zip    /usr/local/.
```

2.- Se descomprime el archivo

```
unzip  jakarta-ant-1.5-bin.zip
```

Se crea la carpeta jakarta-ant-1.5

3.- Se genera una liga a nuestro tomcat

```
ln -s jakarta-ant-1.5  ant
```

y tenemos a jakarta-ant para generar los módulos.

B4. Creación del módulo Webapp

1.- Se descomprime el archivo

```
tar -xvzf  jakarta-tomcat-connectors-4.1.27-src.tar.gz
```

2.- Se ubica a la carpeta de webapp

```
cd jakarta-tomcat-connectors-4.1.27-src/webapp
```

3.- Se ejecuta el archivo de construcción

```
./support/build.sh
```

4.- Se configura habilitando tomcat y apxs de apache

```
./configure -enable-java=/usr/local/tomcat -with-apxs
```

5.- Creamos el modulo con make

```
make
```

6.- Copiamos el módulo a apache

```
cp /apache-2.0/mod_webapp.so /etc/httpd/modules/.
```

Y está listo el modulo para que pueda trabajar apache y tomcat.

B5. Modificación de httpd.conf

1.- Se edita el archivo httpd.conf

```
vi httpd.conf
```

2.- Se modifica el nombre del servidor y se asigna el nombre y puerto, que son muy importantes.

```
ServerName linux.uaq.topico2:80
```

3.- Se cambia el DocumentRoot

```
DocumentRoot "/usr/local/tomcat/webapps"
```

4.- Agregar las siguientes líneas al final del archivo

```
LoadModule webapp_module modules/mod_webapp.so
<IfModule mod_webapp.c>
 WebAppConnection warpConnection warp localhost:8008
  WebAppDeploy examples warpConnection  /examples
 </IfModule>
```

B6. Modificación de Server.xml

1.- Se modifica el archivo Server.xml

```
vi server.xml
```

2.- Se agregan las siguientes lineas

```
<Connector
className="org.apache.catalina.connector.warp.WarpConnector"
port="8008" minProcessors="5" maxProcessors="75"
enableLookups="true" redirectPort="8443"
acceptCount="10" debug="10" connectionTimeout="20000"
useURIValidationHack="false" disableUploadTimeout="true" />
```

3.- Se cambia el nombre del Engine, por el que tiene apache, en éste caso es 148.220.50.94

```
<Engine name="Tomcat-Apache" defaultHost="148.220.50.94" debug="0">
```

4.- Se modifica el nombre del host virtual y la ubicación del directorio base

```
<Host name="148.220.50.94" debug="0" appBase="webapps"
unpackWARs="true" autoDeploy="true">
```

Se comenta el conector que establece un escuchador en el puerto 8080 así se ha configurado Tomcat para trabajar con apache.

10. Bibliografía.

1) JAVA 2 Manual de usuario y tutorial. 3ra. Edicion. Agustin Froufe Alfaomega, Ra-Ma. p. 7-11.
2) JINI code camp chapter 0-4, JINI in action, SUN.
3) JAVA2 Tutorial de Java en línea Froufe http://www.cica.es/formacion/JavaTut/Cap2/metodv.html
4) Tutorial en línea de Juan Antonio Palos, Sun http://www.programacion.com/java/tutorial/servlets_basico/3/
5) Java SUN Technology JSP Overview on line http://java.sun.com/products/jsp/overview.html
6) Java Sun Tecnology Overview http://java.sun.com/products/jdbc/overview.html
7) Java Sun Technology java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html
8) Struts Wikipedia http://es.wikipedia.org/wiki/Jakarta_Struts
9) Manual Linux incluido con la distribución Red Hat 8.
10) "La Biblia de Red Hat Linux 8" Christopher Negus Ediciones Anaya Multimedia.
11) MySQL Technical Reference.
12) Curso de Tomcat http://www.programacion.com/java/articulo/desp_servlets/#estructura
13) JSP Manual de Referencia, Phil Hanna, McGraw-Hill, Primera Edición
14) Manual de Apache en Linux
15) Manual en línea apache www.apache.org
16) Sistemas de Bases de Datos, C. J. Date, Segunda Edición