UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
ARAGON

“INFORME DE UN PROYECTO DE SISTEMA DE REGISTRO DE QUEJAS CON SOFTWARE LIBRE”

TRABAJO ESCRITO PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

POR LA MODALIDAD DE
“SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL”

P R E S E N T A :

YAEN EDUARDO SANDERS MONTOYA.

ASESORA: MTRA. SILVIA VEGA MUYTOY

MÉXICO 2011

INDICE

Introducción.

En el presente trabajo se describe lo aprendido a lo largo de cada uno de los módulos del Diplomado "Desarrollo de Sistemas con Software Libre Linux" y los cursos: "Desarrollo de aplicaciones con *PHP* y *PosgreSQL*", "Introducción al lenguaje *JAVA*" y "Desarrollo de aplicaciones web con *JSPs* y *SERVLETs*". La importancia y relevancia que cada uno de los tópicos se encuentran plasmados de una manera entendible y sencilla, para lograr una mayor comprensión de los temas expuestos.

Objetivo

La finalidad de este trabajo es exponer los contenidos que se vieron a lo largo de los cursos antes mencionados, haciendo un resumen en palabras sencillas que expliquen lo que se aprendió en los mismos, procurando siempre la veracidad de la información y la correcta aplicación de los temas.

Justificación

Los cursos y diplomado citados fueron escogidos por dos motivos fundamentales: el tema que contemplan y la continua capacitación. Los tópicos manejados son de mi interés debido a que en la actualidad son el principio con el que trabajo, además de que la filosofía del software libre es interesante por los alcances que puede tener, la practicidad de operación, el área de aplicación, la seguridad y la confiabilidad que ofrece como sistema operativo, dejando ver ventajas que vale la pena rescatar y aplicar en los sistemas actuales.

En la actualidad con el rápido avance de las tecnologías, métodos y sistemas, es necesario mantenerse actualizado para ofrecer soluciones eficientes, eficaces, seguras y practicas al momento de desarrollar aplicaciones y considero que estos temas ofrecen un panorama que puede ser explotado tomando en cuenta que son las bases de algo de mucho mayor alcance, siendo éstas importantes para actualizarse en temas más específicos en momentos posteriores.

Es por lo anterior que se optó por la selección de estos temas de entre muchos disponibles, por ser de mi interés y por que en el campo laboral están teniendo un auge importante, relevante, generando mayor competencia y mejores resultados.

A continuación se describen de manera breve y general cada uno de los tópicos que serán expuestos en el presente trabajo, para formar una visión panorámica de lo visto durante la capacitación obtenida.

El diplomado "Diseño de Sistemas con Software Libre: Linux" contó con siete módulos divididos en los siguientes temas:

1. **Sistema Operativo Linux**, es el módulo introductorio al Sistema Operativo Linux, la distribución que se manejó fue Slackware, se conocen los comandos básicos para el uso adecuado del mismo.

2. **Instalación y Administración de Linux**, módulo en el que se detalla la instalación y administración del sistema operativo para lograr el más óptimo rendimiento del mismo.

3. **Editores para la creación de páginas WEB**, se describen las principales etiquetas HTML, hojas de estilo (CSS) y la combinación de ambos para lograr una página web.

4. **Administración de Servidores WWW en Linux**, enfocado a la instalación, configuración y seguridad del servidor de páginas web Apache.

5. **Programación con PHP**, instalación y configuración del archivo *php.ini*, comandos principales, creación de funciones, código embebido, optimización de la capa de negocio y una introducción a la Programación Orientada a Objetos.

6. **Interacción de WWW con Base de Datos**, instalación y configuración del manejador de base de datos *MySQL* y conexión entre Apache, *PHP* y *MySQL*.

7. **Introducción a la seguridad en Cómputo**, historia, modelos de seguridad, tipos de cifrados, estrategias y buenas prácticas.

Así mismo, el contenido de los cursos que tomaron como complemento se describe a continuación:

1. **Desarrollo de aplicaciones web con *PHP* y *PostgreSQL***, módulo adicional del diplomado anterior en donde se conocieron los aspectos del lenguaje de programación *PHP* interactuando con el manejador de Bases de Datos, *PostgreSQL*.

2. **Introducción al Lenguaje JAVA**, curso en el que conocen los comandos básicos del lenguaje de programación *JAVA*, instalación, manejo en línea de comando e interacción con Eclipse.

3. **Desarrollo de aplicaciones web con Java JSP's y Servlets**, instalación, administración y funcionamiento del servidor *Glassfish*, así como el desarrollo de *JSPs* y *SERVLETs* así como la configuración del archivo *XML*.

1. Informe General del Diplomado de "Desarrollo de Sistemas con Software Libre en Linux".

El diplomado que a continuación se detalla fue de mi interés dado que representa una alternativa económica y sencilla al desarrollo de aplicaciones, además de las múltiples herramientas que nos permite implementar el software libre, aunado a la creación de sistemas resulta mucho más interesante.

Las ventajas que nos ofrece el sistema operativo Linux, desde mi punto de vista, son determinantes en aspectos de suma importancia al momento de desarrollar aplicaciones, temas tan importantes como la seguridad de los datos, estabilidad del sistema y la administración del mismo son unas de las cartas de presentación más convincentes que ofrece el software libre, de ahí el interés de conocer sus alcances.

1.1 Sistema operativo Linux.

En este módulo se da una explicación de que es el sistema operativo Linux, su manera de trabajar, su estructura de directorios y archivos. Su duración es de 20 hrs. Es un módulo importante ya que su objetivo es el de unificar conceptos en cuanto a Linux.

Para el Sistema Operativo Linux la estructura es arborescente, es decir, desde la raíz (root) se desprenden los demás directorios y subdirectorios, los cuales son:

- **/home**: Será el directorio de los usuarios
- **/var/spool/mail**: Donde se almacenarán los archivos de correo electrónico.
- **/sbin**: Comandos de sistema
- **/mnt**: Unidades montadas temporalmente
- **/tmp**: Almacena archivos temporales
- **/etc**: Almacena algunos archivos de configuración del sistema
- **/root**: Directorio de inicio del usuario root
- **/var**: Se almacenan logs, bases de datos, entre otros

1.1.1 Comandos y utilerías básicas.

Se manejaron y aplicaron los comandos básicos para el manejo de Linux además de cómo integrar dichos comandos para su correcto uso, basándose en la estructura primaria de los comandos en Linux: **comando *argumento1, argumento2,…, argumento n***, algunos ejemplos son:

- **cd**: Cambiar de directorio
- **man**: Es la ayuda en línea de los comandos de Linux.
- **pwd**: Imprime en pantalla el directorio actual de trabajo.
- **touch**: permite crear archivos vacíos de tamaño 0.
- **hostname**: muestra el nombre del servidor.
- **uname**: Indica el tipo de Linux que se está utilizando.
- **ps**: Permite ver los procesos que se están ejecutando

- **ls**: listar archivos en un directorio
- **mkdir**: crear un directorio
- **rm**: borrar archivos

Existen diversas maneras de integrar los comandos en Linux para poder obtener lo que se desea en pantalla, para lograrlo se utilizan caracteres especiales tales como:

- **> <** (mayor o menor que): Sirven para redireccionar la salida de algún comando a otro archivo.
- **|** (pipe): Sirve como una tubería, es decir, redirecciona la salida de comando en la misma línea, no se almacena ni guarda en ningún archivo.
- **>>**: Realiza un redireccionamiento no destructivo o de adición entre los archivos involucrados.

Esta serie de comandos puede ser utilizada con metacaracteres.

1.1.2 Nociones de administración

Dentro de lo que es la administración de Linux, se comentó que dependiendo de las funciones que vaya a desempeñar la máquina en donde se instale son las medidas de administración que se deben tomar; dentro de las más relevantes están si manejará Bases de Datos, correos electrónicos de usuarios, necesidades de puertos como el *ftp* o *ssh*, entre otras consideraciones dependerá el espacio otorgado a cada uno de los directorios que los contienen.

Otro aspecto primordial de la administración es la gestión de usuarios y sus permisos dentro del sistema, éstos se podrán crear, modificar o eliminar.

Este tema nos dio la pauta para introducirnos en el amplio mundo del software libre y su manejo de manera general, mostrando los procesos más comunes, ahora tendremos que ver como instalarlo y como administrarlo para explotar de mejor manera las ventajas que nos ofrece y así utilizarlo eficaz y eficientemente.

1.2 Instalación y administración de Linux.

En este módulo del diplomado se vieron unos de los aspectos más importantes del sistema operativo Linux, como lo son su instalación y administración. Estos dos procesos serán las bases para que podamos desarrollar un sistema estable, seguro y con la disponibilidad que el usuario siempre espera de este tipo de aplicaciones.

La filosofía de Linux esta basada en tres preceptos básicos que definen de manera muy acertada la manera de operar y el modo en que nació este Sistema Operativo:

- Su estructura modular lleva a su filosofía principal "*Entre más pequeño mejor*".
- Permite la reducción de esfuerzo combinando las herramientas.
- Es un sistema personalizable de acuerdo a las necesidades de cada usuario.

Linux (*Linux Is not Unix*) es un sistema operativo compatible con Unix, está inspirado en *Minix* que es un pequeño sistema desarrollado en Unix por Andy Tanenbaum. Linux se distribuye libremente, su filosofía es tomada en base a la **GNU** (*General Public License*), la cual habla acerca de que el código fuente tiene que estar siempre accesible y cualquier modificación debe de ser publicada y compartida a la comunidad.

1.2.1 Perfil y actividades del administrador.

El administrador del sistema es la persona encargada de configurar, mantener y actualizar el sistema o conjunto de sistemas que conforman una red. Cuida que todo el hardware, software y periféricos se encuentren en las condiciones más óptimas para que los usuarios puedan utilizar el sistema sin problemas.

El perfil que debe cubrir cualquier administrador de sistemas está determinado por la capacidad de resolver problemas, así como la correcta aplicación de sus conocimientos en los momentos que se requieren, además debe ser una persona capaz de lidiar de manera pacífica y siempre amigable con los usuarios y sobre todo con un gran sentido de la responsabilidad, aunado a esta lista otros aspectos que debe tener un buen administrador son:

- Planeación de actividades, para seguir un orden en las acciones a tomar.
- Tener siempre copias de seguridad, siempre debe de respaldarse información de los usuarios, bases de datos, correo electrónico y archivos originales del sistema.
- Conocer utilerías básicas del sistema.
- Conocimientos básicos de hardware y mantenimiento de dispositivos.
- Establecer políticas de uso y administración, estas son reglas muy importantes de definir puntualmente, ya que serán las que regulen las acciones que tendrán los usuarios dentro del sistema.

1.2.2 Dar de alta y/o baja del sistema.

El administrador debe conocer las diversas maneras para dar de baja o apagar el sistema, para agilizar procesos y así no interrumpir las tareas de los usuarios. Para esto, hay que comprender el proceso de encendido del sistema, lo primero que realiza el proceso de arranque de Linux es cargar una copia del kernel que se encuentra en el directorio */boot*; después detecta todos los dispositivos de hardware y los pone a disposición del sistema, busca en los scripts de configuración para leer el nivel de inicio configurado por default, luego busca los scripts del *runlevel* y finalmente ejecuta el script *rc.local*.

El nivel de inicio (*runlevel*) por default está definido en el archivo **etc/inittab**, dependiendo del que se configure las rutinas de encendido y apagado pueden variar, se cuenta con 6 niveles.

Ahora bien, para apagar el sistema se pueden utilizar varios comandos, la manera en que éstos apagan o reinician el sistema puede variar por lo que hay que determinar la manera más apropiada:

- **poweroff**, es una manera peligrosa de apagar el sistema, ya que no verifica que los procesos existentes hayan terminado adecuadamente, además da de baja todos los servicios.
- **halt**, es parecido al anterior, sólo que éste llama al comando *halt* quien es el que apaga el sistema.
- **shutdown**, considerado como el más apropiado para apagar y reiniciar el sistema ya que lo hace de una manera eficiente y puede configurarse para que se realice en un tiempo determinado, permitiéndole a los usuarios guardar su información.
- **Init**, si se desea cambiar el modo de ejecución del sistema se puede hace con este comando seguido del *runlevel* deseado.

1.2.3 Mantenimiento de claves de usuarios

Una de las tareas de la administración, es la del manejo de las claves de los usuarios del sistema, es importante tener en cuenta las altas, bajas y modificaciones de éstas, así como definir perfectamente los grupos a los que van a pertenecer los usuarios siempre tomando en cuanta la capacitación de los mismos para generar contraseñas fuertes, asignar permisos adecuados y la administración de los recursos para evitar el abuso de ellos.

1.2.4 Sistema de archivos.

Los sistemas de archivos son la forma en que el sistema operativo organiza, gestiona y mantiene la jerarquía de los archivos en los dispositivos de almacenamiento, generalmente discos duros. En Linux existen diversos tipos de sistemas de archivos, cada uno con características diferentes que determinan la funcionalidad del sistema ante fallos de software, hardware o eléctricos. Se determinan al momento del formateo de particiones en la instalación del sistema; existen varios con ventajas y desventajas pero todos bajo el estándar del ext2, dependerá del usuario y del propósito que se le de al sistema para realizar la mejor elección.

En Linux se clasifican en tres categorías:

- Basados en disco (ext2, ext3, ReiserFs, XFS, JFS, etc.).
- Sistemas remotos de red (NFS, Coda, Samba, etc.).
- Sistemas especiales (procfs, ramfs y devfs).

Ext2.

Este es el sistema de archivos del cual se desprenden los demás. Incluso hasta nuestros días sigue siendo utilizado por más usuarios de Linux, dentro de sus ventajas se encuentran:

- Solidez.
- Fácil actualización.
- No utiliza FAT sino i-nodos.
- Cuenta con corrección y detección de errores.

Ext3.

Este es el tercer sistema de archivos extendido, es parecido al Ext2 sólo que aquí se implementan las transacciones. La idea de las transacciones es la de grabar las acciones antes de llevarlas a cabo efectivamente, resultando un sistema de archivos coherente siempre, con verificaciones rápidas y recuperaciones eventuales; por lo que el tiempo que toma verificar un sistema de archivos ya no es proporcional al tamaño del mismo sino al uso verdadero que se hace del mismo.

Sus ventajas y características principales son:

- Disponibilidad.
- Integridad de los datos.
- Velocidad.
- Fácil migración.

ReiserFS.

A diferencia de los anteriores este sistema de archivos fue elaborado desde cero, es transaccional como Ext3, pero su estructura es radicalmente diferente ya que esta basada en árboles binarios. Ofrece una gran estabilidad y un tiempo de arranque del sistema muy corto después de una caída del mismo. Tiene una eficacia media en el estado de los datos después de una caída súbita y no cuenta con herramientas de recuperación de datos.

1.2.5 Instalación y actualización de Linux (Slackware).

En el diplomado se instaló la distribución de Linux Slackware y la instalación que se realizó fue mediante CD-ROM, para la ejecución de este sistema operativo no se requiere de un equipo muy poderoso ya que los requisitos mínimos son:

- Procesador 386.
- Memoria RAM de 16 a 64 MB.
- Espacio en disco de 500 MB a 3 GB.
- Floppy.
- Unidad de CD-ROM.

Se tratará de explicar de manera sistemática y breve el proceso de instalación de Slackware, se cuenta con 3 discos, los primeros 2 son el sistema operativo y sus librerías y el último es de material extras, como las de idioma entre otras.

Al momento de insertar el primer disco aparecerá una pantalla de bienvenida a la instalación de Linux y en la parte inferior de la pantalla el indicador *boot:* en donde solamente se dará *Enter,* al hacer esto se cargan automáticamente las imágenes de los discos *boot* y *root*, también detectará la mayor parte del hardware instalado en el equipo. Una vez que se realiza este proceso el primer paso es el de configurar el teclado, al ser un teclado en español el que se utilizará se elige la opción *qwerty/es.map* de la lista que se despliega en pantalla.

Ahora se debe probar la localización de los caracteres en el teclado para verificar que la distribución de los caracteres del teclado corresponda con el tipo de mapa que se eligió. Para cambiar de mapa, se teclea el número 2, y para aceptar la configuración del teclado presionamos el 1.

Es momento de registrarse ante el sistema para que asigne un *shell* y poder continuar con la instalación, para esto hay que registrarse como el superusuario de Linux, *root*.

Ya que el sistema asignó nuestro *shell*, se puede trabajar para continuar con este proceso, llegó la hora de preparar el disco duro y determinar la distribución de sus particiones, es importante hacer hincapié que se requiere de un esquema de disco bien definido dependiendo de la funcionalidad que tendrá el servidor, de esto dependerá la facilidad de administración y flexibilidad del sistema. Lo más recomendable es hacer particiones dependiendo de la carga de archivos de los usuarios, los servicios, etc. Para hacer más fáciles y eficientes las tareas de respaldo y actualización de Linux.

Para particionar un disco se utiliza el comando *dev/hda*, para los discos con arquitectura *IDE* y *dev/sda* para los que sean *SATA*. Cuando se ejecuta el sistema mostrará una pantalla donde se tendrá que decidir que se quiere hacer, en caso de no saber, se utilizará la ayuda del comando *fdisk* presionando la letra *m* para que despliegue las opciones disponibles.

Para una instalación básica sólo se crearán dos particiones la *SWAP* y la nativa de Linux. Para crear una partición nueva se escribe la letra *n*, desplegara un menú donde se tienen dos tipos de particiones primarias o extendidas, se crea una extendida escribiendo la letra *e*.

Ahora se tiene que definir el tamaño de la partición, el sistema ofrece por default el primer cilindro disponible del disco, para indicar el final de la partición se utiliza un método de suma de mega bites a partir del punto de inicio de la partición, esto es, si se desea una partición de 258 Mb, solo tendremos que escribir *+258M*. El espacio que se acaba de definir corresponderá al área de *SWAP*.

Se define a continuación la partición nativa de Linux, en donde se tendrá instalado el sistema operativo, para realizarlo se siguen los mismos pasos que la anterior, escribiendo la letra *n* para crear una nueva partición y posteriormente se define su tamaño, en este caso como se quiere que ocupe el resto de los cilindros disponibles en el disco duro, sólo se dará *Enter* cuando pida este dato y por default ocupará los cilindros restantes.

Es conveniente verificar si se crearon correctamente las particiones, para esto se escribe la letra *p* y se verifica el disco.

Una vez que se concluye con este proceso y se verifica que se hayan ejecutado correctamente las instrucciones ingresadas, es el momento de escribir en el disco esta información y se hace tecleando la letra *w*, aparecerá un mensaje de aviso de que se alteró la tabla de particiones y regresará al *shell* de Linux. Ya que se guardan los cambios en las particiones es momento de comenzar con la instalación con el comando *setup*.

La instalación ahora es en formato gráfico y lo primero que se tiene que definir es el *KEYMAP* como se hizo anteriormente desde el *shell*. Realizando este proceso y el asistente detectará automáticamente la partición *SWAP* para que se le de formato y se prepara para

ser utilizada, por lo que selecciona la opción *Yes*, y al darle formato se elige *Ok*. Ahora muestra la partición de Linux para que se realice el mismo procedimiento que con la anterior, por lo que se selecciona la partición en donde se va a instalar el sistema operativo y se formatea. Una vez que tenga formato la partición de Linux, se selecciona el sistema de archivos que más convenga para el sistema operativo.

El siguiente paso es determinar desde que recurso se va a realizar la instalación, en este caso será el de CD. Para seleccionar los paquetes que se quieran instalar se utiliza la barra espaciadora para elegir los que se necesiten. Por facilidad hay que elegir la instalación completa (*full*).

Después hay que decidir cual *kernel* es el que se va a instalar, esto dependerá de la arquitectura y características de la computadora donde se esta instalando. Ahora el asistente ofrece la posibilidad de crear un disco de arranque para el sistema, el cual puede servir como recuperador en el momento que falle.

El siguiente paso es habilitar el *hotplug*, esta opción reconoce gran cantidad de dispositivos de hardware y ayuda a la activación automática de la mayoría de ellos.

Llega el turno de instalar el gestor de arranque *LILO*, el cual ayuda a generar un menú de inicio de los sistemas operativos que se tienen instalados en el equipo, debido a que en esta instalación están contemplada dos sistemas operativos se prosigue a configurarlo, para ello se selecciona el modo experto.

Lo recomendable para instalar *LILO* es hacerlo en la *MBR* del disco duro, después preguntará en que disco se desea instalarlo, esto para el caso de que se tengan más unidades instaladas, en caso contrario se elige el que aparece en automático. Una vez seleccionada esta opción se prosigue a agregar los sistemas operativos que se tengan instalados. Se tiene que elegir la partición del disco en el que se encuentra alojado el sistema, así como la etiqueta que se desea sea visible en el menú inicial. Para finalizar con la instalación del *LILO*, solo hay que elegir en el asistente que lleve a cabo esta tarea.

También se tiene que configurar el tipo de mouse con el que se cuenta, hacerlo es un proceso muy sencillo, sólo seleccionar el tipo y listo.

Ahora viene algo muy importante, la red, lo primero es determinar el nombre del equipo o el *hostname,* después el dominio en el que se encuentra este equipo, si es que se tiene; luego determinar el tipo de IP con la que va a identificarse la máquina en la red. Finalmente entrega un reporte de todo lo que se acaba de configurar, suponiendo que todo es correcto, se continúa al siguiente paso.

Se pueden configurar los servicios que se desea estén disponibles en el sistema, sólo se tiene que seleccionar en la pantalla que presente el asistente gráfico. Determinar la interfaz gráfica con la que iniciará Linux cuando así sea requerido, posteriormente se le dirá que se desea reemplazar el archivo *etc/fstab*.

Para finalizar con la instalación sólo se tiene que escribir una contraseña segura para el usuario *root*, y reiniciar el equipo eligiendo la opción EXIT, en este momento el equipo saca la charola del CD y con la combinación de teclas *Ctrl.+Alt+Supr*. Cuando inicie nuevamente el

sistema se verá el gestor de arranque *LILO* y en este momento se tiene instalado satisfactoriamente Linux, ahora sólo queda configurarlo de la manera que se necesite.

1.2.6 Instalación y administración de dispositivos.

En la administración de dispositivos, Linux siempre los nombre de tal manera que siempre pueda tener control de ello y reconocerlos fácilmente, iniciando por los discos duros instalados que están en el directorio */dev*, y las unidades serán nombradas dependiendo como estén conectados al equipo, generalmente se le asignan las letras del abecedario en para ordenarlos de mayor a menor según su jerarquía, por ejemplo, *dev/hda*, será el disco maestro, *dev/hdb* el disco esclavo, y así sucesivamente con las demás unidades.

Una de las herramientas que hace a Linux un sistema ordenado y fácil de usar es el hecho de que las unidades se montan, esto quiere decir que cuando se conecta algún dispositivo externo se puede elegir en que directorio lo queremos tener, sin embargo existe un directorio que se crea automáticamente para este fin, */mnt*, aquí se pueden tener todos los elementos que se quieran. Para montar un dispositivo se ocupa el comando *mount*, cuya sintaxis básica es:

mount (dispositivo) [*tipo*] (punto de montaje)

Una de las ventajas de este sistema operativo es que aparte de poder montar y desmontar dispositivos, tiene la ventaja de que se pueden tener particiones montadas permanentemente, esto es, por ejemplo si tienes dos sistemas operativos instalados.

1.2.7 Administración del área de SWAP.

La memoria *swap*, es un espacio en disco duro en el cual se almacenan procesos que se están ejecutando en tiempo real para ser captados por el procesador, es una ayuda a la memoria RAM. En Linux se puede destinar una partición completa como *swap*, también se pueden crear archivos destinados a ser utilizados como tal, cabe aclarar que no es de ninguna manera un sustituto de la memoria *RAM*. Es recomendable que sea el doble de la memoria *RAM* que se tiene en el equipo.

1.2.8 Monitoreo del sistema.

El objetivo de monitorear el sistema es el de administrar los recursos del mismo evitando tener procesos que estén ocupando una gran parte de la capacidad del procesador o que estén generando archivos que saturen el disco duro.

En Linux se ejecutan diversos servicios o programas que son traducidos en procesos; al proceso que puede generar otros procesos se le *llama proceso padre*.

Hay 5 tipos de procesos en Linux:

- **Preparado.** Esta listo para ejecutarse.
- **Ejecutando.**
- **Suspendido.** Se encuentra esperando algún tipo de evento.

- **Detenido.**
- **Zombie.** Cuando un proceso es hijo, y dicho proceso se termina avisa al proceso padre de esto, si por algún motivo el proceso padre no lo elimina se dice que se queda en este estado.

Se pueden detener o eliminar procesos utilizando el comando *Hill***.**

1.2.9 Realización de respaldos.

Una buena política de administración en Linux son los respaldos, ya que se previenen pérdidas de datos. Linux permite hacer paquetes de archivos, los cuales pueden comprimirse para que no ocupen mucho espacio. El comando para hacer respaldos es *tar*, cuya sintaxis es:

tar –*zcvf* archivo.tar.gz (*archivo o directorios a empaquetar*)

Con el parámetro *z* se le indica que lo comprima, con *c*, que lo cree, *v* permite visualizar en pantalla el proceso de compresión y finalmente *f*, indica el nombre del archivo.

1.2.10 Interfaces gráficas en Linux.

En Linux existen diversas interfaces gráficas, las más conocidas son GNOME Y KDE, esta última estándar de Linux. Ambas incluyen aplicaciones gráficas de administración, diseño, desarrollo y de ayuda para la configuración de las herramientas y servicios de Linux, propician un ambiente más amigable para el usuario ahorrando tiempo en muchas operaciones.

En esta parte también se puede configurar el teclado y mouse, toda la configuración de la interfaz *X* de Linux se encuentra en el archivo *etc/X11/xorg.conf*, donde se pueden encontrar las configuraciones anteriores entre otras.

1.2.11 Configuración y servicios básicos de la red.

En estos tiempos la red es un elemento básico para la computadora, para configurarla en Linux es muy sencillo y se puede hacer de dos maneras: desde línea de comandos o por medio del asistente gráfico que se muestra al ejecutar el comando *netconfig*. Ambas formas harán su trabajo eficientemente, siendo más recomendable la primera.

Dentro del directorio */etc* hay dos archivos que ayudan con la configuración de la red:

- **hosts**. Contiene la dirección *IP* del equipo relacionada con su nombre, si se estuviera en una red privada en este archivo se asociaría la dirección *IP* del equipo y el nombre de dominio.

- **resolv.conf.** En este archivo se guardan las direcciones *IP* de los servidores de nombres *DNS*, incluso aquí se pueden agregar manualmente estos elementos.

Cuando se realizan cambios a estos archivos es necesario reiniciar el servicio de red para que surtan efecto los mismos, esto se hace de la siguiente manera:

/etc/init.d/networking restart

Dependiendo de las características del equipo, se tienen diversas interfaces de red (dos o más tarjetas de red, red inalámbrica, entre otras). Para saber el estado de cada una de ellas, se usa el comando *ifconfig*.

Para configurar manualmente una red, es decir, desde la línea de comandos hay dos sintaxis que serán útiles, la primera para agregar una dirección *IP* fija a la tarjeta:

#ifconfig eth0 (*dirección IP*) netmask (*dirección IP de la máscara de red*) broadcast (*dirección IP del broadcast*) up

Y para configurar el Gateway o puerta de enlace:

#route add default gw (*dirección IP del gateway*)

1.2.12 Introducción a la seguridad.

Linux es un sistema operativo que funciona autónomamente excelente, sin embargo es común que sea utilizada con servicios de red, ya sea para administrarlos o para generarlos, esto lo hace vulnerable, debiendo además, ofrecer protección contra todos los usuarios y a sí misma.

En todo buen sistema operativo la seguridad es básica, se deben contemplar aspectos como el envió y/o recepción de datos, dentro y fuera del sistema o usuarios mal intencionados que puedan comprometer la estabilidad del sistema.

Existe dentro de la seguridad informática y la administración de sistemas el llamado *triángulo dorado de la seguridad*, el cual consiste en 3 conceptos básicos que todo administrador debe conocer e implementar en los sistemas que maneje:

- **Integridad**. La información sólo puede ser modificada por quien está autorizado y de manera controlada.

- **Confidencialidad**. La información debe ser accesible únicamente para los usuarios autorizados.

- **Disponibilidad.** Debe estar disponible siempre que se necesita.

La seguridad contempla aspectos como son el análisis de riesgos, el cual es un proceso en el cual se estudian las vulnerabilidades de los sistemas y amenazas a las cuales están expuestos. Primero hay que hacer un inventario del equipo que se tiene, el servidor o máquina, la estructura de la red, el edificio donde se encuentra ubicado, entre otras cosas.

Teniendo estos aspectos totalmente definidos, se procede a identificar las amenazas a las cuales se está expuesto, se suele dividir las amenazas en dos grandes grupos:

- **Naturales**, terremotos, huracanes, incendios, etc.

- **Ocasionados por el hombre,** virus informáticos, ataques maliciosos, intercepción de paquetes, ingeniería social, entre otros.

Ante este tipo de amenazas se deben de tomar acciones de tipo preventivo, de manera que se puedan hacer planes emergentes para contrarrestarlas mientras el sistema se vuelve seguro, confiable y estable.

Como se pudo apreciar este módulo representa la columna vertebral del funcionamiento del sistema operativo, de ahí su importancia; ahora que ya tenemos la idea de cómo instalarlo y como llevar una mejor administración del mismo, podemos comenzar a ver un poco sobre el como diseñar los sistemas, comenzando con lo más elemental, el HTML.

1.3 Editores para la creación de páginas web.

La presentación siempre es importante en cualquier proyecto y situación, es por eso que en este apartado se describe como utilizar y conjugar los elementos principales para poderle dar una mejor visualización a nuestras aplicaciones mediante el Lenguaje de Marcas de Hipertexto (HTML).

Por sus siglas en ingles **HTML** (*Hyper Text Markup Language*) es el lenguaje de marcas de hipertexto. Esto quiere decir que es un lenguaje de procesamiento de texto para que sea interpretado por algún navegador o *browser* haciendo uso de ciertas marcas o etiquetas, es decir, les dice a los navegadores como deben de mostrar el contenido de una página web.

Un *browser* (navegador de Internet) es una aplicación que permite al usuario visualizar páginas web elaboradas con HTML. Algunos ejemplos de navegadores son:

- Firefox.
- Opera.
- Google Chrome.
- Netscape.
- Safari.
- Internet Explorer.

El origen de HTML se remonta a los años ochentas, cuando el físico Tim Berners-Lee, trabajador de la Organización Europea para la Investigación Nuclear propuso un sistema de hipertexto para compartir documentos. Tras finalizar su sistema, y después de unirse con Robert Caullin, presentan la propuesta **WWW** (*World Wide Web*).

Como se mencionó éste es un lenguaje caracterizado por marcadores o etiquetas, de las cuales existen dos tipos en HTML: abiertas (*<etiqueta>*) y cerradas (*<etiqueta> </etiqueta>*)

La estructura básica de un documento HTML se divide en tres etiquetas principales:

- **<html></html>** que le indica al navegador que todo lo que se encuentre entre estas etiquetas será código HTML.

- **<head></head>**, entre estas etiquetas se encontrará el encabezado del documento, existen varios elementos que pueden encerrarse entre estas etiquetas como los estilos para los elementos del documento *<link>*, referencias a archivos que se ocupen *<script></script>*, metadatos de ubicación *<meta>*, el título que se mostrará en el navegador *<title></title>*, entre otros.

- **<body></body>**, dentro de esta etiqueta se escriben todos los componentes que formarán la página web, tablas, listas, imágenes, formularios, etc. Es el cuerpo del documento HTML.

1.3.1 Organización del Texto.

Una de las partes importantes de la presentación es la organización de lo que estamos presentando y los detalles que puedan ser distintivos para marcar temas o hacer énfasis en alguna parte de la información, describiremos algunos elementos para lograrlo.

1.3.1.1 Saltos de línea y párrafo

La distribución del texto en una página web es, en ocasiones, lo que le da vista al trabajo, de ahí la importancia de utilizar las etiquetas adecuadas para darle formato al escrito.

Si se quiere realizar un salto de línea sencillo se utilizará la etiqueta *
*; mientras que si se desea escribir un párrafo, el contenido se escribirá entre las etiquetas * .*

1.3.1.2 Separadores

El separador es una línea horizontal que sirve para diferenciar entre los diferentes elementos de la página. Su etiqueta es *<hr>* y tiene atributos como:

- **align**. Determina la alineación ya sea a la izquierda, derecha o centrado.
- **width**. Determina la longitud, ya sea en pixeles o en porcentaje, de la línea.
- **size**. Especificará el ancho de la línea y su valor se escribe en pixeles.

1.3.2 Estilos del texto y fuentes

La orientación, tamaño y características del texto son relevantes para lograr la atención del usuario, para que al momento en que se encuentre con él, le sea fácil de comprender visualizar y entender, veamos algunas de las características básicas de presentación del texto.

Estilos físicos

Sirven para cambiar la apariencia del texto, como son:

- **. Texto en negritas**
- **<s></s>.** ~~Tachar Texto~~
- **.** *Texto en cursivas*
- **<u></u>**. <u>Texto subrayado</u>.

Estilos lógicos

Los estilos lógicos sólo sirven para enfatizar ciertas partes del texto:

- **.** Muy similar a las negrillas
- ****. Similar a las cursivas, enfatiza el texto.
- **<code></code>.** Para indicar código fuente de un programa
- **<cite></cite>**: Palabras textuales.

Etiqueta <FONT>

Esta etiqueta permite modificar los atributos del tipo de letra que se están utilizando como son:

- **face.** Tipo de letra (arial, comic sans, MS sans Serif). Pueden incluirse varias por si los distintos navegadores no pueden interpretar dicho tipo de letra, cuando es así van separadas por comas.
- **color**. Define el color del texto y puede recibir valores hexadecimales o en texto de colores primarios pero en inglés.
- **size**. Define el tamaño de la letra, va del 1 al 7, por default se toma el 3.

Las cabeceras o títulos son otra manera de trabajar el texto sin necesidad de utilizar la etiqueta *<font>* ya que ayudan a mostrar los títulos en diferentes niveles de importancia, teniendo como valores del 1 al 6 de menor a mayor. Estas cabeceras están definidas por las etiquetas *<h1>* a la *<h6>*, siendo la primera la de mayor tamaño.

1.3.3 Listas

La organización del texto para que sea claro a los lectores de páginas web es importante, es por esto que existen las listas y hay una manera de definirlas mediante etiquetas *html*. Existen tres tipos de listas: marcadas, numerada y de definiciones.

Listas Marcadas

Las listas marcadas sirven para mostrar información agrupada sin ningún orden o secuencia. La etiqueta para definirlas es *
* y los elementos se indican con *
*. Su estructura es:

```
<ul>
      <li>Elemento1</li>
      <li>Elemento2</li>
      <li>ElementoN</li>
</ul>
```

Se puede indicar el tipo de viñeta dentro de la etiqueta *<ul>* con su atributo *type*, el cual recibe los valores de *disc* (circulo relleno, este valor es el default), *circle* (circulo sin relleno) y *square* (cuadrado relleno).
Listas numeradas

Se definen entre las etiquetas *
*, su estructura es la siguiente:

```
<ol>
      <li>Uno</li>
      <li>Dos</li>
      <li>Tres</li>
</ol>
```

Se puede definir la secuencia de la lista que se va a usar con el atributo *type* y recibe el atributo *start*, para definir a partir de que elemento va a comenzar la lista.

Listas de Definiciones

Este tipo de listas son ideales para escribir conceptos con sus definiciones, su estructura es la siguiente:

```
<dl>
      <dt>Concepto1
      <dd>Definición de Concepto1
      <dt>ConceptoN
      <dd>Definición de ConceptoN
</dl>
```

1.3.4 Hipertexto

El hipertexto es un vinculo o referencia hacia otro documento o archivo HTML mediante una liga, son útiles para la intercomunicación entre páginas o mostrar archivos.

- Referencias Locales
- Referencias a documentos HTML externos
- Imágenes

Tablas

Las tablas se utilizan para mostrar de manera más ordenada el contenido de la página web, para hacerla más entendible y para que la lectura se le facilite a los usuarios. Teniendo básicamente: tablas básicas y avanzadas de acuerdo a la información que se maneje

Frames

Son elementos en desuso actualmente, debido a que ya existen elementos más vistosos para suplirlos como lo es la creación de menús interactivos; sin embargo pueden llegar a ser útiles en algún momento del diseño de aplicaciones, es por esto que se muestran sus elementos e implementación. Son una serie de marcos los cuales son independientes dentro de la ventana que se está usando, pueden cargarse y acomodarse de distintas formas.

Para utilizar *frames* se tiene que hacer uso de dos etiquetas, *<frameset>*, para definir la estructura de las secciones en las que se dividirá la pantalla y *<frame>*, para determinar los archivos que se publicaran en cada una de las secciones definidas con la etiqueta anterior.

La etiqueta *<frameset></frameset>* determinará la distribución de las divisiones independientes que existirán dentro de la ventana.

Etiqueta <FRAME>

Con la etiqueta *<FRAME>* se le indica al HTML cuales son los archivos y en que parte de la distribución principal deberán mostrarse. Algunos de sus atributos son:

1.3.5 Hojas de estilo

Los estilos pueden indicarse directamente en los distintos elementos del documento HTML, desafortunadamente si se hace de esta manera y se desea cambiar varios documentos, se tendrá que hacer el cambio uno a uno los documentos a modificar. Es por eso que se usan las hojas de estilo, que son archivos que contienen la morfología de los diferentes atributos de los componentes de los archivos HTML.

Etiqueta <STYLE>

Con la etiqueta *<style></style>* se definen de manera interna en *HTML* los estilos que se emplearán en el documento. Esta etiqueta tiene un atributo muy importante el cual es *type***.** Su valor es *text/css*; ya que con éste se le dice al código *HTML* que lo interprete como estilo. Existen tres tipos de estilos:

- **Interna.** Se define con la etiqueta *<style>* y siempre va entre las etiquetas *<head></head>.*

```
<style>
   p {
     font-family:Arial;
     font-size:12pt;
      color:red;
      background-color:lime;
       }
   p.otro {
     font-family:Times;
     font-size:13pt;
     color:blue;
     background-color:yellow;
     }
</style>
```

- **Externa**. Este tipo de estilo es el más recomendable, ya que se conforma de un archivo independiente en el cual se definen todos y cada uno de los estilos que se utilizarán, y en el documento *HTML* sólo hay que incluir este archivo, el cual tiene que guardarse con extensión ***css***, escribiéndolo de la manera siguiente

```
parrafo {
     font-family:Times;
     font-size:13pt; color:blue;
     }
```

Una vez hecho esto, se incluye en el documento *HTML* el archivo *CSS* de la siguiente manera:

```
<link rel="stylesheet" href="miestilo.css">
```

- **Directo en etiqueta.** Es el método más simple, todas las etiquetas tienen un atributo *style* al cual le podemos dar valores de la siguiente manera:

```
<p style="font-family:Times; font-size:13pt;"></p>
```

1.3.6 Formularios

Los documentos HTML tienen la ventaja no sólo de mostrar información, sino que en conjunto con otros lenguajes de programación vía web (php, asp), pueden ser de gran utilidad para interactuar en diversos sistemas web.

- La etiqueta <*input*>, Define la mayor parte de los elementos dentro de un formulario.

- La etiqueta <*textarea*>, sirve para escribir textos grandes como lo son descripciones u observaciones, se puede determinar el tamaño de columnas y renglones que tendrá.

- La etiqueta <*select*>, se utiliza para hacer listas de selección al usuario, dependiendo de sus atributos es que se pueden seleccionar una o más opciones.

- Las etiquetas <*option*> son las que contendrán los valores que se mostraran en el *select*. Su único atributo es VALUE, el cual contiene el valor de la opción seleccionada, si tiene la leyenda *SELECTED* esa será la opción seleccionada por *default*.

1.3.7 Introducción a XML

XML son las siglas en inglés de *Extensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el *W3C*. Es una simplificación y adaptación del *SGML* y permite definir la gramática de lenguajes específicos de la misma manera que *HTML* es a su vez un lenguaje definido por *SGML*. Por lo tanto *XML* no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que lo usan para su definición son XHTML, SVG, MathML.

Historia de XML

XML fue desarrollado por un grupo de trabajo bajo los auspicios del consorcio World Wide Web (W3C) a partir de 1996. Este fue constituido en 1994 con el objetivo de desarrollar protocolos comunes para la evolución de Internet. Se trata de un consorcio de la industria internacional con sedes conjuntas en el Instituto Tecnológico de Massachussets, de Estados Unidos, el Instituto Nacional de Investigación en Informática y Automática Europeo y la Keio University Shonan Fujisawa Campus de Japón. El W3C tiene como misión la publicación para uso público de protocolos o estándares globales de uso libre.

1.3.8 Conceptos básicos de XML

El objetivo del *XML* es almacenar y transmitir datos. Estos documentos se componen de elementos *XML*, cada uno de los cuales consta de una etiqueta de inicio, de una fin y de los datos comprendidos entre ambas etiquetas. Al igual que los documentos HTML.

Analizadores de XML

Un analizador *XML* es un programa el cual puede leer este tipo de archivos y obtener sus datos facilitando el acceso a las diversas aplicaciones con las cuales se quiere interactuar. De acuerdo a la versión de *XML* que se este utilizando, será el analizador que se use, pero *HTML* incluyo un analizador único para interpretar *XML*.

Un **DTD** (*Document Type Definition* o Documento de Tipo de Definición), es la mezcla de la utilización de un documento XML como fuente de definición de datos y el muestreo de dichos datos mediante otro aplicación. Por un lado, se tienen los datos bien definidos y en el frente se tiene la aplicación que interpreta y muestra esos datos.

Crear y usar DTD's

Primero se tiene que elaborar un archivo que será el que almacene el *DTD* y luego se creará un documento *XML*. El documento resultante se puede abrir con un navegador de Internet y se podrá ver el contenido ordenado de la lista de personas.

El diseño será un elemento fundamental, pero para que las aplicaciones orientadas a web funcionen correctamente deben contar con un servidor dedicado para este fin, los mas indicados son los servidores WWW, estos ofrecen opciones interesantes a la hora de la implementación de sistemas, su instalación, administración e implementación se describirá en el siguiente punto.

1.4 Administración de servidores WWW en Linux.

Un servidor *www* o servidor web es un programa que esta diseñado para transferir hipertextos, páginas web o páginas *HTML*: textos complejos con enlaces, formularios, botones, objetos incrustados como las animaciones, etc. Esta transferencia la logran mediante el protocolo *HTTP* y la noción de sitio virtual.

Hay diversos servidores web, pero con el que se trabajó en el diplomado fue el servidor Apache, el cual es un software libre multiplataforma, que en sus inicios consistía únicamente en parches a aplicar al servidor de NCSA. Apache cuenta con bastantes elementos altamente configurables como lo son los mensajes de error, bases de datos de autenticación y negociado de contenido, sin embargo ha sido criticado por la falta de una interfaz gráfica que haga más amigable su configuración; aún así es el servidor web más utilizado debido a su gran eficiencia. Además este servidor se desarrolló dentro del proyecto *HTTP Server* (*httpd*) de la Apache Software Foundation.

Funcionamiento de Apache.

El funcionamiento del servidor Apache en Linux es sencillo, lo hace de la siguiente manera:

- Se ejecuta el programa de apache en forma de demonio (programa que atiende peticiones de servicios), el cual es ejecutado por root y por omisión levanta 5 procesos hijos. El demonio o proceso padre se encarga de administrar a los hijos.

- Cuando un usuario teclea una dirección iniciando con *http://* se determina que usará el puerto asignado en los archivos de configuración antes mencionados.

- Después manda una petición al *DNS* para que resuelva el nombre de la página tecleada para que regrese al equipo solicitante la dirección *IP* a la que corresponde la dirección ingresada. Una vez que resuelve cual es la dirección *IP* solicitada regresa la petición por la red al equipo solicitante para que sea atendida por el puerto configurado.

- Al llegar la petición al servidor, se identifica en la información que se solicita el protocolo *http* y la solicitud del puerto configurado, por lo que el demonio *httpd* asigna a uno de sus procesos hijo para que atienda la petición, buscando la información solicitada y enviándola a la dirección *IP* de donde vino la petición inicial.

- Cada proceso hijo atiende a la vez a una petición, de tal forma que al recibir la petición de información, el demonio *httpd*, asigna a un proceso hijo que atiende la petición y envía el o los archivos que se solicitan y continua atendiendo la petición de información desde el inicio hasta el fin del envío de todos los archivos solicitados. Una vez que termina de atender la petición el proceso hijo se libera y queda nuevamente a la espera de que se le asigne una nueva petición.

- El puerto del servicio, el número de procesos hijos y la mayoría de los parámetros del servidor apache pueden ser configurados según las necesidades del administrador.

Esquemáticamente se puede representar su funcionamiento como se muestra en la Figura 1.4.1.

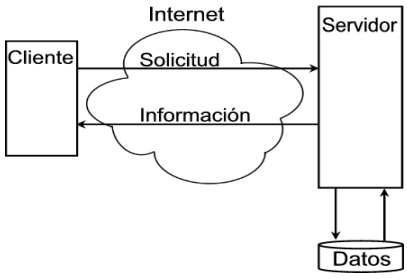


Figura 1.4.1 Representación gráfica del funcionamiento de los servidores WWW como Apache.

Instalación del servidor www.

La instalación de Apache es sencilla en este diplomado se utilizó a versión 2.0.63 del servidor y para tenerla en el equipo sólo hay que seguir los siguientes pasos.

- Descargar el código fuente de la sección de downloads de la página http://httpd.apache.org/download.cgi.

- Descomprimir los archivos descargados con los siguientes comandos:

  # **gzip** -d httpd-2_1_63.tar.gz
  # **tar** xvf httpd-2_1_63.tar

- Ahora hay que configurar la estructura de directorios para la plataforma donde se va a instalar, se hace mediante el comando:

  # **./configure** - -*prefix* = (*ruta de instalación*).

  El parámetro *prefix* se puede omitir y la instalación quedaría en el directorio */usr/local/apache.*

- Ahora se compilan las diferentes partes de apache con el comando *make*.

- Instalar el paquete en el directorio especificado con *prefix*, en caso de haberlo hecho, o en el automático si no se hizo, esto con el comando *make install*.

- Personalizar el servidor editando el archivo *prefix/conf/httpd.conf.*

- Existe en la instalación un script de control de Apache alojado en el directorio de los binarios (*/bin*) de Linux, servirá para levantar o parar el servidor y también para probar que la instalación funcione, haciéndolo así:

  # *PREFIX/bin/apachectl start*

- Una vez que se inicia el servicio del apache y para poder usarlo se deben guardar todos los archivos que necesitemos publicar en el directorio *PREFIX/htdocs.*

- Para detener el servicio lo se hace de la siguiente manera:

  #PREFIX/bin/apachectl stop

1.4.1 Configuración del servidor.

Que nuestro servidor funcione como se desea depende de la configuración que se le haga, se pueden hacer varias cosas interesantes con Apache, el archivo principal de configuración es *httpd.conf* que se encuentra en el directorio donde se tenga instalado el servidor web. He aquí las directivas configurables más interesantes de este archivo.

- **ServerRoot**. Es el directorio principal o raíz a partir del cual se colocan los archivos de configuración, error y bitácora.

- **Listen**. Permite especificar una dirección *IP* o puerto para que el servidor atienda peticiones, el valor por default es el puerto 80.

- **Configuración del grupo y usuario de apache**. Se puede especificar el nombre del grupo de usuarios que tendrán acceso al servidor así como el nombre del usuario que lo puede ejecutar, sólo se escribirá **Group** (*nombre del grupo*) y **User** (*nombre del usuario*).

- **ServerAdmin**. Es parte de la configuración principal del servidor, aquí se especifica el correo electrónico del administrador para que cuando ocurra algún error Apache se lo notifique por este medio.

- **DocumentRoot**. Es el directorio donde se van a encontrar alojadas todas las páginas web que se necesiten publicar, organizadas en carpetas para tener un mayor control para acceder a ellas.

- **DirectoryIndex**. Especifica el o los archivos con sus extensiones que Apache mostrará si el directorio principal es solicitado por algún usuario.

- **ErrorLog**. Se puede configurara el archivo en el que se guardará la bitácora de errores del servidor, se recomienda crear el archivo de la ruta especificada en esta directiva.

- **ErrorDocument**. Aquí se especifica la ruta del archivo al que se redireccionará Apache en caso de que exista algún error, los más comunes son: ligas rotas (*error 404*), error interno del servidor (*error 500*), etc.

1.4.2 Accesos restringidos.

El acceso por contraseñas es un proceso que permite verificar quien entra a los directorios de las páginas web del servidor, para entrar a un sitio protegido se solicita un usuario y una contraseña que se configuran previamente.

Si se tiene información que deseas que sólo tengan acceso un reducido número de usuarios, una de la soluciones puede ser una directiva del script de configuración de apache, *htaccess.*

Hay dos maneras de realizar la configuración, una directamente en el archivo *httpd.conf*, suponiendo que el directorio de acceso restringido esta ubicado en */usr/local/apache/passwrd/secreto*, la configuración de la directiva se tendrá que hacer en la directiva

<Directory /usr/local/apache/passwrd/secreto>

O bien crear un archivo dentro del directorio protegido llamado *.htaccess*, el cual va a contener las siguientes directivas:

AuthType Basic
AuthName "*Comentario para mostrar de que es restringido el acceso*"
AuthBasicProvider file

AuthUserFile "*ruta del archivo con los passwords*"
Requiere user *nombre del usuario*

1.4.3 Control de tareas.

El servidor web Apache se puede configurar para atender un mayor o menor número de solicitudes por parte de los usuarios, esto puede ser útil cuando se tienen servidores donde están alojadas varias páginas web con la finalidad de optimizar el proceso de respuesta de la solicitud.

Es por esto que Apache tiene un archivo donde se puede configurar lo anterior, ubicado por default en la ruta */etc/httpd/extra/httpd-mpm.conf*, aquí se pueden ajustar el rendimiento del servidor ya que permite definir el número de procesos hijo o servidores con los que iniciará el demonio *httpd*, así como establecer el número máximo de peticiones que atenderá simultáneamente.

1.4.4 Manejo de sitios virtuales.

Otra de las virtudes de este servidor web es que permite manejar sitios virtualmente, esto es, se pueden tener varios sitios en el servidor y se puede configurar para que el acceso a ellos parezca que estén en otra parte, definiendo los parámetros principales específicos de cada sitio, como el correo electrónico del administrador, el directorio principal donde se guardarán sus archivos así como el nombre del sitio y sus alias, por mencionar los más importantes.

Para lograr esto hay que agregar la siguiente línea al archivo de configuración principal *http.conf*:

include /etc/httpd/extra/httpd-vhosts.conf

Después se tiene que localizar el archive que se encuentra en la siguiente ruta */etc/httpd/extra/httpd-vhosts*, el cual se encarga de configurar todo lo relativo a los sitios virtuales.

Apache no registra automáticamente a los servidores *DNS*, es necesario darlos de alta, para hacer pruebas locales se pueden agregar los nombres en el archivo *hosts*.

Ahora que ya sabemos como sacar un mejor provecho a la configuración el servidor Web, veamos que es lo que se puede lograr cuando lo conjugamos con un lenguaje de programación dinámico como lo es el PHP.

1.5 Introducción al lenguaje PHP

PHP (*Hypertext Pre-Processor*) es un lenguaje de programación interpretado, diseñado para la creación de páginas web dinámicas. Está basado en el modelo *cliente-servidor* el cual consta de tres partes: la primera es del lado del cliente, el cual hace una petición al servidor, php la analiza se la manda al servidor, el cual después de todo un proceso ejecuta una

respuesta que es tomada nuevamente por *PHP* y mostrada al cliente en la etapa de visualización.

PHP fue creado en 1994 por Rasmus Lerdorf, el cual se baso en *Perl* y *CGI's* binarios para la transferencia de datos vía web. La versión más reciente y estable de *PHP* es la versión 5, pero se anuncia que en breve saldrá la versión 6. Dentro de las muchas ventajas de la programación en *PHP* es su orientación a objetos, su portabilidad, conexión con múltiples bases de datos, uso de extensiones, es de código libre basado en la licencia *GLP*, no requiere definición de tipos de variables, entre otras.

1.5.1 Herramientas elementales

Para que se pueda explotar al máximo la utilización de *PHP*, hay que tener en cuenta ciertos requerimientos:

- Un servidor Web con la capacidad de analizar e interpretar el código *PHP*, para este caso se usará el servidor de páginas web apache 2.0.

- Un manejador de base de datos para el cual tenga soporte *PHP*, como lo es *MySQL*, *PostgreSQL*, *Oracle*.

- En el archivo *php.ini* se encuentra toda la configuración de *PHP*, su ubicación dependerá de la instalación y la cual se tendrá que especificar en el archivo *httpd.conf* de Apache.

Directivas de PHP.

PHP es un lenguaje interpretado, para que el navegador sepa que se está utilizando código PHP, se hará uso de las etiquetas:

- **<? ?>:** Etiquetas cortas
- **<?php ?>:** Etiquetas largas.

Hay muchas sentencias útiles en PHP, entre otras las de control.

1.5.2 Common Gateway Interface.

Las aplicaciones *CGI* fueron creadas para el contenido dinámico de las páginas Web. En estas aplicaciones el cliente ingresa ciertos datos a través de un formulario el cual es preanalizado y preprocesado para que el servidor termine de hacer estas tareas, arroje un resultado y sea mostrado al cliente. Un poco como la arquitectura *cliente-servidor*.

Como diseñar una aplicación CGI.

Las aplicaciones *CGI* son muy útiles en los motores de búsqueda o en los registros a través de formularios. *PHP* ofrece una serie de apoyo en estas tareas muy grande. Para diseñar un *CGI*, primero tendrá que hacer:

- Un formulario en *html* con los diversos campos que se necesiten.
- Las interfaces en código PHP para validar los datos.
- Una vez validados, se ingresan a una base de datos, si es el caso, sino, se procede a mostrar los datos.

1.5.3 PHP y la programación orientada a objetos.

La programación orientada a objetos (*POO*) es una metodología en la cual los sistemas se modelan creando clases, que son definiciones a partir de las cuales se crean objetos. Una clase es la definición de algún objeto, conformado de atributos que lo hacen único, además de que las clases se forman de métodos, acciones que pueden realizar los objetos. A partir de la distribución de *PHP5* se puede incluir la programación orientada a objetos (*POO*).

La clase class.

PHP permite definir clases con la palabra reservada *class,* su implementación es sencilla y la manera de hacer referencia a ella también, la estructura básica de una clase en *PHP* sería definida de la siguiente manera:

```
<?php  class humano {
/** Definición de los atributos de la clase humano*/
             var $ojos;
             var $pies;
             var $manos;
             var $boca;

/**Constructor*/
             function humano(){
             }

/**Definición de métodos */
             function hablar(){
             }
}?>
```

Como se pudo ver este lenguaje de programación permite elaborar aplicaciones web de una manera sencilla y dinámica, sin embargo siempre esta la posibilidad de almacenar datos en los sistemas para hacerlos más útiles, por lo que necesitamos una base de datos donde alojar todo lo que deseamos revisar posteriormente, por lo que en el siguiente apartado describiré lo que se vio al respecto en este diplomado.

1.6 Interacción de WWW con Bases de Datos.

Las bases de datos son elementos primordiales en estos tiempos por el valor que tiene la información dentro de las empresas, recordando la importancia de tenerla siempre a la mano y de manera comprensible para cualquiera, por lo que son parte fundamental de la mayoría de los sistemas que se desarrollan actualmente.

Dentro de todo sistema es necesario guardar la información generada ya que servirá como referencia o como un dato de consulta según pase el tiempo. Para lograr el almacenaje de información existen las bases de datos, las cuales son un conjunto de información organizada y relacionada, dentro de un archivo lógico. *SQL* (Lenguaje de Consulta Estructurado), es el estándar para las bases de datos relacionales, creado en 1981 por IBM y esta compuesto por comandos, cláusulas y funciones, y la combinación de estos elementos permite la creación, actualización y manipulación de todos los objetos de la base de datos.

En 1995 surge en Suecia un sistema de gestión de bases de datos relacional, cuyo objetivo era el de cumplir con los estándares de *SQL* sin sacrificar la velocidad de ejecución, la fiabilidad de los datos y la usabilidad del producto, es así como MySQL AB crea *MySQL*. Este nuevo gestionador de bases de datos es uno de los más populares en nuestros tiempos debido a las características que lo componen además de ser *freeware* también genera bases de datos relacionales, muy útiles en las aplicaciones actuales.

La idea de las bases de datos relacionales se basa en crear una representación de la realidad a través de objetos llamados entidades y las relaciones entre ellas. Es por esto que este tipo de bases de datos representan de forma gráfica mediante un **DER** (*Diagrama Entidad - Relación*). Una *relación* es la asociación que existe entre dos entidades, y existen 4 tipos principales:

- **1 : 1**. Uno a uno.
- **1 : M**. Uno a muchos.
- **M : 1**. Muchos a uno.
- **M : M**. Muchos a muchos.

Las bases de datos tienen elementos esenciales que las integran, en los cuales se pueden apoyar para organizar y relacionar la información que se meta en ellas, sus componentes básicos son:

- **Dato**. Es la unidad básica de almacenamiento en una base de datos.
- **Campo**. Es el componente que agrupa a los datos a partir de una misma característica.
- **Registro**. Clasifica a los datos haciéndolos independientes y diferentes uno del otro.
- **Archivo**. Elemento lógico que almacena los datos en forma de tablas.

Todos estos elementos son primordiales para lograr los objetivos primordiales de las bases de datos, como son: el disminuir las redundancias, mantener la consistencia e integridad de los datos, así como el de tratar de lograr la máxima seguridad y acceso a los mismos cuando sea necesario.

1.6.1 Tipos de datos y tipos de tablas.

Existen elementos que hay que ir elaborando para poder tener finalmente una base de datos armada, uno de los principales elementos son las tablas, ya que éstas contendrán los datos que se van a guardar y utilizar. Existen varios tipos de tablas:

- **MyISAM**. Tipo de tabla predeterminada, sirve para las necesidades del usuario medio y es compatible con todos los campos y parámetros.

- **ISAM**. No se puede utilizar entre sistemas operativos y no procesa datos mayores a 4 GB.

- **HEAP**. Se utiliza por su velocidad, no admite *auto_increment* ni campos de tipo *blob* o *text*.

- **InnoDB**. Ideal para aplicaciones de gran tamaño, cuenta con un mecanismo de bloqueo de filas para que el usuario no modifiquen o añadan la misma fila a una tabla.

- **BDB**. Tiene las mismas características que la anterior, sólo que se ocupa para aplicaciones de mayor tamaño.

Una tabla esta compuesta por campos, éstos almacenan datos y pueden ser de diversos tipos según las necesidades del usuario o dependiendo de la información que se guarde en las tablas, los datos se engloban en tres grandes categorías:

- Carácter. *(char, mediumtext, mediumblob* y *text*).
- Numéricos. *(tinyint, smallint, mediumint, int, bigint, float* y *double)*
- Otros. *(date, time, year, timestamp o datetime)*

1.6.2 SQL intermedio.

El manejador estándar de bases de datos, *SQL*, tiene sentencias y comandos que son básicos para poder manejar una base de datos. Es primordial el saber manejar las tablas y los datos que en ellas se almacenan para obtener los resultamos que se desean.

Manejo de tablas.

Existen diversos comandos que ayudarán a obtener el control en la creación, modificación y destrucción de las tablas que conformen la base de datos. La Tabla 1.6.1 muestra las principales sentencias y la acción que producen.

SENTENCIA	ACCIÓN
DESCRIBE *Nombre_Tabla*	Muestra la estructura de la tabla *Nombre_Tabla*.
SHOW CREATE TABLE *Nombre_Tabla*	Muestra como fue creada la tabla *Nombre_Tabla*.
DROP TABLE *Nombre_Tabla*	Elimina la tabla *Nombre_Tabla*
ALTER TABLE *Nombre_Tabla*	Altera la definición de la tabla *Nombre_Tabla*
CREATE TABLE *Nombre_Tabla*	Crea la tabla *Nombre_Tabla*, aquí se pueden definir los campos que tendrá así como el tipo de dato del que se trate.

Tabla 1.6.1 Sentencias principales de SQL para manejo y administración de Bases de Datos.

Existen atributos de las tablas que son muy importantes de definir y que serán de utilizad al momento de consultar la información que se guarde en ellas, las llaves, y existen 2 tipos:

- **Primarias** (*Primary Key*). Es el atributo o conjunto de atributos elegidos para identificar un elemento de la tabla de forma única, existen las llaves compuestas, si es que la llave primaria tiene más de un atributo.

- **Foráneas** (*Foreign Key*). Se encarga de relacionar las entidades y está representada por la llave primaria de otra tabla, útiles para la integridad referencial.

Sentencias más importantes en SQL.

Existen muchos comandos para ejecutar en *SQL*, sin embargo existen unos que en definitiva cualquiera que desee manejar una base de datos debe de conocer a la perfección, ya que les ayudarán a poder llevar acabo una buena administración de las mismas. Entre las sentencias básicas se encuentran:

- **SELECT**. Consulta registros de una tabla siguiendo criterios establecidos en la sentencia. Para el complemento *WHERE* se pueden utilizar diversos operadores de comparación entre campos para lograr obtener el resultado que se desea.

- **INSERT**. Comando utilizado para ingresar registros en la base de datos

- **UPDATE**. Actualiza uno o más registros de la base de datos, según los criterios establecidos en la sentencia.

- **DELETE**. Elimina uno o más registros.

1.6.3 Creación de usuarios y permisos.

El crear usuarios en *MySQL* es útil determinar las acciones que tendrán sobre la base de datos y los privilegios para ejecutar sentencias sobre las tablas de la misma. Existen 2 maneras estándar de crear usuarios:

- **GRANT**. Este comando creará un usuario al mismo tiempo que se puede definir los privilegios que tendrá sobre una determinada base de datos.

- **CREATE USER**. Con esta función seguida del usuario que se quiere agregar, se inserta uno nuevo, solo que esta vez no tendrá asignados los privilegios de uso, para asignárselos se ejecutará el comando anterior.

Por lo anterior se puede deducir que el mejor método para la creación de usuarios y asignación de privilegios es el primero, y como se puede leer en su sintaxis existen varios permisos que se pueden dar al usuario que se esta creando, dentro de éstos se tienen:

- **ALL**, concede todos los permisos.
- **ALTER**, permiso para alterar tablas.
- **CREATE**, permiso para crear tablas.
- **DROP**, permiso para eliminar tablas.
- **INSERT**, permiso para agregar registros.
- **DELETE**, permiso para eliminar registros.

- **SELECT**, permiso para realizar la consulta de registros.
- **UPDATE**, permiso para modificar registros.
- **INDEX**, permiso para crear índices.
- **SHOW DATABASES**, permiso para ver todas las bases de datos.

Así como se asignan permisos pueden quitarse los mismos con sólo utilizar el comando *REVOKE.*

1.6.4 Respaldos.

La utilización de los respaldos para las bases de datos es una práctica importante e imprescindible ya que evita perdidas de información y un soporte fidedigno de lo que se tiene. Para realizar un respaldo en *MySQL* se realizan con el comando *mysqldump* de la siguiente manera:

```
mysqldump --all-databases > respaldo.sql
mysqldump --databases bd > respaldo.sql
```

También se pueden restaurar las bases de datos, es útil a la hora de hacer alguna migración de equipos a partir de un respaldo, con sólo utilizar el comando *mysql* ejecutándolo desde el shell de *MySQL* de la siguiente manera: **`mysql`** *`base_de_datos < respaldo.sql.`*

1.6.5 Interacción de PHP y MySQL.

Las bases de datos por si solas son útiles, sin embrago se requiere de cierto conocimiento de la materia para poder manejarlas adecuadamente, es por esto que en la actualidad se desarrollan aplicaciones donde las bases de datos son el motor de alimentación de lo que se muestra en pantalla a los usuarios, uno de los lenguajes de programación que permite esto y que se tuvo como tema en el diplomado fue PHP. Este es un lenguaje comúnmente utilizado en páginas web, ya que su manejo dinámico permite realizar aplicaciones interesantes.

Instalación de MySQL y PHP

El primer paso para llevar a cabo la instalación es descargar los códigos fuentes de ambas aplicaciones, una vez que se tienen en disco, se procede a instalar *MySQL*, con los siguientes pasos, ejecutados desde una consola de Linux:

- **tar –zxvf mysql-5.xxx.tar.gz**, descomprime el archivo descargado.
- **cd mysql-5.xxx**, cambia al directorio donde lo descomprime.
- **./configure --prefix =/usr/local/mysql**, configura la estructura de directorios donde se va instalar, en el atributo *prefix* se puede configurar una ruta específica de instalación.
- **make**, compila los archivos a instalar.
- **make install**, realiza la instalación en el directorio especificado con *prefix*.
- **groupadd mysql,** se crea el grupo que utilizará el gestionador.
- **useradd –g mysql mysql,** se agrega el usuario *mysql* al grupo con el mismo nombre.
- **cd /usr/local/mysql,** cambia al directorio de instalación.

- **/usr/local/mysql/bin/mysql_install_db --user = mysql,** dar de alta el usuario que se acaban de crear en los binarios de ejecución de *mysql*.
- **chown –R root, chown –R maysql var,** establece como dueño del directorio de *mysql* al usuario *root*.
- **bin/mysqld_safe --user mysql,** levanta el servidor de *mysql*.
- **bin/mysql,** ejecuta el servidor de bases de datos, es decir, entra al cliente de *MySQL*.

Con el listado anterior ya se tiene trabajando el servicio del gestionador de bases de datos, ahora hay que instalar las librerías de *PHP* como módulo dinámico para poder utilizarlo y que, a través de él, se pueda manejar los datos almacenados. Para su instalación se siguen estos pasos:

- **tar –zxvfphp-5.xxx.tar.gz,** descomprimir el archivo que se descargó.
- **cd php-5.xxx,** cambiar al directorio de descompresión.
- **make clean,** asegurarse de que se eliminen las versiones anteriores que se tengan instaladas.
- **./configure,** configurar la estructura de directorios.
- **make,** compilar.
- **make install,** instalar.
- Agregar las siguientes líneas al archivo de configuración de *PHP* que se encuentra ubicado en */usr/local/php5/lib/php.ini*:
  - **AddType application/x-httpd-php .php .phtml**
  - **AddType application/x-httpd-php-source .phps**
- Verificar que exista en el archive php.ini la líea que dice: **LoadModule php5_module modules/libphp5.so**

Para verificar que la instalación de *PHP* como módulo dinámico se realizó correctamente, se crea un archivo que se llame *info.php* en la carpeta donde se publican las páginas web y que se configuró con Apache (*/usr/local/apache/htdocs*) con el siguiente código:

```
<?php phpinfo(); ?>
```

Ahora se reinicia el servicio de Apache y en un navegador se teclea la siguiente dirección: http://localhost/info.php, si aparece la información correspondiente al PHP que se acaba de instalar, se tiene todo listo para utilizar en conjunto Apache, *PHP* y *MySQL*.

Funciones PHP para interactuar con MySQL.

La interacción de los módulos que se acaban de instalar es una mezcla interesante, y más aún si se pueden manejar la base de datos desde un script en *PHP*, esto es posible gracias a que existen funciones que lo permiten, las más importantes son las mostradas en la Tabla 1.6.2.

FUNCIÓN	SINTÁXIS	DESCRIPCIÓN
mysql_connect	**mysql_connect(***$equipo, $usuario, $password***)**	abre una conexión a la base de datos ingresando los datos requeridos en los parámetros de la función
mysql_select_bd	**mysql_select_bd(***$basedatos,$conexion***)**	Selecciona la base de datos con la que se va a trabajar.
mysql_close	**mysql_close(***$conexion***)**	si la conexión no es persistente la cierra.
mysql_affected_rows	**mysql_affected_rows(***$conexion***)**	Devuelve el número de filas afectadas por la última operación ejecutada.
mysql_create_db	**mysql_create_db(***$basedatos, $conexion***)**	Crea una base de datos
mysql_db_query	**mysql_db_query(***$basedatos, $consulta, $conexion***)**	Ejecuta una consulta.
mysql_error	**mysql_error(***$conexion***),**	Devuelve un mensaje de error si lo hubiera, en la última operación ejecutada.
mysql_field_name	**mysql_field_name(***$resultado, $num_campo***)**	Devuelve el nombre del campo especificado del resultado de la consulta.
mysql_field_table	**mysql_field_table(***$resultado, $num_campo***)**	Devuelve el nombre de la tabla del campo indicado del resultado de la consulta.
mysql_list_field	**mysql_list_field(***$basedatos, $tabla, $conexion* **)**	Obtiene los campos de una tabla como resultado de una consulta.
mysql_num_fields	**mysql_num_fields(***$resultado***)**	Devuelve el número de campos del resultado de una consulta.
mysql_num_rows	**mysql_num_rows(***$resultado***)**	Devuelve el número de filas del resultado de una consulta.
mysql_insert_id	**mysql_insert_id(***$conexion***)**	Devuelve el identificador generado para un campo de tipo *AUTO_INCREMENT* generado por el último I*NSERT*

Tabla 1.6.2 Principales funciones de PHP para manejar Bases de Datos en MySQL.

Podemos concluir entonces que las bases de datos son un conjunto de tablas ordenadas lógicamente donde almacenaremos los datos que nos interesen. Una vez diseñado nuestro modelo y ya desarrollado la parte visual y de funcionamiento de nuestra aplicación, solo nos falta uno de los aspectos importantes en los sistemas, la seguridad.

1.7 Introducción a la seguridad en cómputo

La seguridad en cómputo se define como todas aquellas reglas, técnicas o procedimientos destinados a proteger y resguardar toda la información y recursos de relevancia para una empresa, negocio o individuo de cualquier riesgo o peligro que llegará a alterar su funcionamiento directo o los resultados que se obtengan de éste.

Para que un sistema pueda adquirir el titulo de seguro debe de cumplir con los siguientes puntos:

- **Integridad**: La información sólo puede ser modificada por quien está autorizado y de manera controlada.
- **Confidencialidad**: La información sólo debe ser legible para los autorizados.
- **Disponibilidad**: Debe estar disponible cuando se necesita.
- **Irrefutabilidad**: el usuario no puede negar cualquier acción sobre el cambio de su información o la de otros dependiendo el caso.
- **Autenticación**: debe existir un método de logueo para los usuarios.

1.7.1 Vulnerabilidades, riesgos y Amenazas.

Una amenaza es un evento que puede desencadenar un incidente el cual produce daños materiales o pérdidas de alto impacto. Los riesgos son los múltiples factores en los que se puede caer cuando no se tiene una buena base de defensa ante las posibles amenazas, mientras que las vulnerabilidades son las posibilidades de caer en una amenaza.

Existen dos clasificaciones para las amenazas:

- **Lógicas.** Cualquier tipo de programa o software que puede dañar nuestro equipo creado de forma mal intencionada o no (virus, gusanos, troyanos, adware, spyware, rootkits, exploits, dialers, cookies, phishing, spam).

- **Físicas.** Amenazas provenientes de la naturaleza o de los propios medios como el hombre mismo (terremotos, huracanes, incendios, ingeniería social).

Para cada tipo de amenaza hay que contar con un plan de contingencia con el fin de prevenir cualquier daño, ante las amenazas físicas no hay más que hacer planes emergentes, ya que es imposible contener un desastre natural; el objetivo de la seguridad en cómputo es el de ofrecer una respuesta rápida en el mínimo de tiempo posible.

1.7.2 Criptografía.

Es una técnica para cifrar y descifrar información mediante operaciones especiales las cuales se emplean para intercambiar información que no se desea que sea identificada por otra persona. El cifrado se basa en hacer uso de una clave, la cual es una información secreta con la cual se encubre la información, y sólo con ayuda de ésta puede descifrarse el mensaje o contenido.

Las dos técnicas más sencillas de cifrado son:

- **Sustitución.** Se utilizaba en las civilizaciones más antiguas, la cual se atribuye a Julio Cesar; consiste en sustituir las letras de un documento por la tercera letra que le corresponde en el alfabeto; si deseamos escribir una A, escribiremos una D, y así sucesivamente.
- **Transposición.** Consiste en reordenar los datos para cifrarlos a fin de hacerlos ininteligibles, pero geométricamente, un ejemplo de ello es enrollar una tira de papel, escribir el mensaje, cuando esta tira sea desenrollada, el mensaje no tendrá coherencia, pero si se vuelve a enrollar, se podrá descifrar.

Sistemas de cifrado.

Dentro del mundo de la seguridad informática existen varias técnicas para lograr esconder o disfrazar el contenido de los archivos para que no sean vulnerados por otras personas. Todos esos mecanismos de cifrado tienen sus ventajas y desventajas, aquí solo se expondrán como una opción más de la amplia gamma de posibilidades que se ofrece.

Simétrico.

Son aquellos que usan la misma clave para cifrar y descifrar un documento. El principal problema con este sistema es que se debe de encontrar un canal seguro para el intercambio de la clave.

Asimétrico.

También llamados sistemas de cifrado público, usa dos claves diferentes, una es la pública la cual se puede enviar por cualquier medio y la otra es privada, la cual debe guardarse para que nadie tenga acceso a ella. El remitente usa una clave pública para cifrar el mensaje, una vez que es enviado éste, el receptor puede descifrar el mensaje con su clave privada.

Híbrido.

Hace uso de ambos sistemas, funciona mediante el cifrado de la clave pública, para compartir una clave para el cifrado simétrico; en cada mensaje, la clave simétrica utilizada es diferente por lo que si un atacante pudiera descubrir esta clave, sólo le valdría para ese mensaje y no para los restantes. El destinatario usa su clave privada para descifrar la clave simétrica y acto seguido ésta última la utiliza para descifrar el mensaje.

1.7.3 Administración básica de la seguridad.

Ya que Linux ofrece un sistema multiusuario real, muchos de los ataques que se presentan es por querer adueñarse de un usuario normal, una vez que esto sucede se procederá a querer obtener los privilegios de *root* o conseguir su contraseña.

El archivo */etc/passwd* contiene la información de los usuarios, a que grupos pertenecen y sus contraseñas cifradas, mientras que en el archivo */etc/group* contiene el listado los grupos y su identificador.

Permisos.

Uno de los principios básicos en la administración de Linux es el del manejo de permisos en los archivos y/o directorios. El comando para cambiar los permisos en Linux es *chmod.*

Los permisos de un archivo son tres: lectura, escritura y ejecución; y éstos se dividen en tres niveles: el usuario, el grupo y otros.

Para determinar el nivel de permisos se sigue una regla simple, cada permiso tiene un valor, el cual, puede ser sumado y aplicado en los tres niveles (Tabla 1.7.1).

PERMISO		VALOR
Read	r	4
Write	w	2
Execute	x	1

Tabla 1.7.1 Cuadro de valores para determinar los permisos de los usuarios sobre los directorios del sistema.

Entonces sumando los tres dígitos de cada grupo se obtendrá:

El primer dígito (*propietario*) 4+2+1 = 7
El segundo dígito (*grupo*) 4+0+0 = 4
El tercer dígito (*otros*) 0+0+1 = 1
Se sugiere que los directorios que son de suma importancia para el correcto funcionamiento del sistema, no puedan ser vistos por los demás usuarios.

1.7.4 Ataques.

Un ataque es un método por el cual un individuo mediante un sistema informático intenta tomar el control, desestabilizar un sistema y a su vez dañarlo. Muchos de estos ataques suelen ser organizados por *hackers*.

1.7.4.1 Hacker.

La definición de *hacker* tiene un significado un tanto ambiguo, es un usuario el cual tiene un conocimiento amplio en redes, Linux o cualquier sistema operativo y tiene la habilidad de buscar defectos y puertas traseras para entrar a sistemas obteniendo información atacando las vulnerabilidades de éstos.

En la historia han existido *hackers* que debido a sus proezas informáticas, mal intencionadas la mayoría, pero no por eso de alta eficiencia, muy reconocidos y conocidos a nivel mundial, entre los pioneros se pueden encontrar a:

- **Jonathan James.** Sentenciado a sus 16 años ya que instalo un *backdoor* en un servidor de la Agencia de Reducción de Amenazas de la Defensa (*DRTA*) de Estados Unidos, el cual le permitió ver correos electrónicos, nombres de usuario y contraseñas.

- **Adrian Lamo.** Se destaca por sus intrusiones a Microsoft y al periódico The New York Times.
- **Kevin Mitnick.** En un principio uso las redes telefónicas saltando de una a otra para lograr tener la clave de la *PC* personal de Tsutomu Shimomura, el cual era un *hacker* de sombrero blanco dedicado a investigar y rastrear a los usuarios maliciosos; lanzo ataques a empresas tan conocidas como Apple, Motorola o Qualcomm, entre otras.
- **Kevin Poulsen.** Tambien conocido como "*Dark Dante*", se ganó reconocimiento cuando *hackeó* las líneas telefónicas de la radio de Los Ángeles "*KISS FM*", con lo cual obtuvo ganancias para comprarse un Porsche, entre otras cosas. Las fuerzas del orden lo apodaron "*El Hannibal Lecter del crimen informático*".
- **Robert Tappan Morrison.** Hijo de un científico de la Agencia Nacional de Seguridad, y conocido como el creador del Gusano *Morris*. Morrison escribió el código del gusano cuando era estudiante de Cornell University. Su intención era usarlo para ver que tan largo era Internet, pero el gusano se replicaba excesivamente, haciendo las computadoras demasiado lentas.

1.7.5 Seguridad en red.

La seguridad en las redes es un aspecto sumamente importante, ya que la mayoría de los ataques pueden ser llevados a cabo por este medio. En Internet circulan muchos virus que pueden dañar la integridad del servidor.

Una buena protección para el servidor es usar un *firewall*l o muro de fuego (Figura 1.7.1), que es una protección encargada de filtrar los paquetes de datos que entran y salen del servidor.

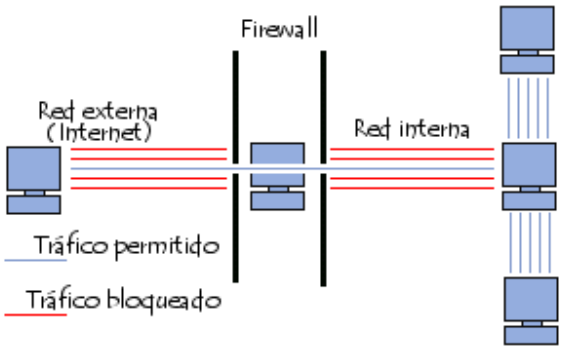


Figura 1.7.1 Modelo básico de la estructura de un *Firewall*.

Hay dos tipos de *firewalls*, los físicos y los lógicos, los primeros son dispositivos que se conectan entre el servidor y la red, cada uno de ellos tiene su configuración distinta. Los lógicos se instalan en el mismo servidor, allí se administran y configuran, en Linux uno de los más famosos es el *IPtables*. Éste es un sistema de cortafuegos vinculado al kernel de Linux el cual tiene una serie de reglas de paquetes, de entrada, salida, y paso de paquetes.

Para ofrecer una seguridad completa en las redes, se puede instalar un software escucha como *Snort*, el cual es un sistema de detección de intrusos, su manera de trabajar es estar captando todo el tráfico de la red asociando las peticiones al servidor con su *IP*, puede imprimir reportes gráficos vía web, si se detectará con *Snort* que una *IP* esta haciendo

muchas peticiones al servidor, mediante *IPtables* se puede decirle que niegue las peticiones a éstas.

Finalmente, la seguridad es de los aspectos más importantes para el correcto funcionamiento de los sistemas ya que garantiza la integridad de los datos que interesan. Existen distintas maneras de integrar las aplicaciones, hasta el momento se ha descrito como sería la integración de un sistema mediante el lenguaje de programación *PHP* y el manejador de bases de datos *MySQL*, sin embargo existen más posibilidades al respecto, en el siguiente capítulo se describirá la manera en que se pueden combinar *PHP* y *PostgreSQL* para la base de datos.

1.8 PHP y PostgrSQL.

En este módulo se conocerá otro manejador de bases de datos como lo es *PosgrSQL* y su interacción con el lenguaje de programación *PHP* parta el desarrollo de sistemas: por la robustez que ofrece este manejador se convierte en otra de las posibilidades a considerar al momento de implementar aplicaciones.

1.8.1 Implementación y configuración del entorno de desarrollo.

En este punto es cuando se van a configurar todos los programas necesarios para el funcionamiento del gestionador de bases de datos *PosgreSQL*, para lo cual se va a necesitar instalar de la misma manera que para *MySQL*, un servidor web, Apache, un lenguaje de programación para realizar el manejo de la información e interfaces, *PHP* y finalmente *PosgreSQL*.

La configuración e instalación de los elementos que se necesitan se realizará de la misma manera que se describió en los puntos respectivos a los servidores wer y a PHP, sólo que en este último al momento de compilarlo, va a hacerlo de la siguiente manera:

./configure --with-apxs --with-pgsql

Esto creará una biblioteca compartida con *libphp4.so* que será cargada por Apache usando una línea *LoadModule* en el archivo de configuración *httpd.conf*, incorporando el soporte para *PosgreSQL* en la biblioteca mencionada.

La instalación de *PosgreSQL* se realiza de la siguiente manera:

- Descargar el código fuente de *PostgreSQL*.
- **installpkg postgresql-8.3.3-i486-1sl.tgz**. Instalar el manejador en el equipo.
- Crear el directorio donde se va a instalar la base de datos inicial, por ejemplo: *mkdir /home/db/pgsql.*
- **chown pgsql /home/db/pgsql**. Asignar el directorio que se acaba de crear al propietario de *PosgreSQL* que regularmente es *pgsql*.
- **su – pgsql**. Para utilizar el manejador de bases de datos es necesario hacerlo como el usuario con los permisos para ejecutarlo, el cual por default en la instalación es el usuario *pgsql* que pertenece al grupo del mismo nombre.

- **initdb -D /home/db/pgsql**. Instalar con el usuario *pgsql*, la base de datos inicial en el directorio que se creó anteriormente. El parámetro *–D* permite definir el directorio donde se quiere instalar la base de datos.
- **posgresql -D /home/db/pgsql**. Iniciar el servicio de *PosgreSQL*.
- **pgsql –l**. Ver las bases de datos que se tienen.
- **psql -d *labasededatos* -U *elusuario***. Para poder tener acceso a las bases de datos sólo lo se hace de esta manera, definiendo la base a la que se quiere entrar y el usuario con el permiso para hacerlo.

1.8.2 Templates en PHP.

Las plantillas en la programación son una alternativa para facilitar el diseño y optimizar la programación, el uso de ellas depende de cada programador y de su viabilidad al momento de la implementación.

Definición y uso.

La idea de trabajar con plantillas o *templates*, es la de separar la programación de la aplicación del diseño gráfico, esto para que en un futuro cuando se quieran hacer cambios a cualquiera de las dos partes del proyecto no se necesite mover más que lo que nos interesa.

Las plantillas contienen archivos con el código HTML que conforma todo el molde del sitio, como el cuerpo, los encabezados y el pie de página entre otros elementos, esta es la parte gráfica y de visualización, en donde se definirán algunas variables propias del *template* que después serán manejadas por la aplicación; por otro lado está el código de la aplicación, que será la encargada de realizar las consultas a la base de datos y manejar la interfase por medio del motor de *templates* sin preocuparse del diseño.

Ventajas y desventajas.

La utilización de plantillas brinda grandes beneficios como los siguientes:

- Independencia entre la aplicación y la interfase, que es lo más recomendado en todo tipo de software.
- Rediseño del sitio, sin necesidad de tocar el código fuente de la aplicación.
- Mayor facilidad de actualización del sitio.
- El mantenimiento del código es más sencillo, ya que no te preocupas por el *HTML*.

Como se puede apreciar son bastante convincentes las ventajas para utilizar *templates* a la hora de diseñar una aplicación, pero como en todo, hay inconvenientes, pero si los sabes balancear ambas partes puedes tomar la mejor decisión a la hora de diseñar un sitio o aplicación web. Dentro de las desventajas de las plantillas están:

- La programación es más pesada.
- Su tiempo de procesamiento puede hacer caer el rendimiento del sitio.

La clase Class.NokTemplate.

El *template* que se utilizó en el curso fue *NokTemplate*, una clase que implementa la idea de plantillas a la programación, es una de las clases más utilizadas para este tipo de casos.

Las características principales de funcionamiento de esta plantilla son:

- **Modo Debug**, posibilita la optimización de los scripts.
- Cuenta con un **sistema de cahé temporal en archivo externo**, incrementando la velocidad de procesamiento.
- Utiliza expresiones compatibles con Peral mucho más veloces, ya que tiene **velocidad de interpolación**.
- Es compatible con ***FastTemplate*** ya que utiliza la misma lógica.
- Soporta definición de bloques anidados dentro de las plantillas.

La lógica para manejar esta plantilla es sencilla, simplemente hay que seguir estos 4 pasos que son básicos y fundamentales:

- Cargar Template.
- Asignar valores a las variables, las cuales se definen entre llaves, por ejemplo *{Mi_Variable}*, también son sensibles al uso de mayúsculas y minúsculas.
- Analizar el contenido al expandir el template.
- Mostrar los resultados.

Este *template* es bastante dinámico en cuestión de implementación, ya que el diseño se presta para este fin, ver el ejemplo más sencillo que constará de 2 plantillas: *cuerpo.html* (Tabla 1.8.1) y *contenido.html* (Tabla 1.8.2).

cuerpo.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html> <head>
<title>{TITULO}</title>
</head>
<body>
        {CONTENIDO}<br>
</body>
</html>
```

Tabla 1.8.1 Contenido del archivo cuerpo.html en la implementación de *NokTemplate*

contenido.html

```
Hola, mi nombre es {NOMBRE}.
```

Tabla 1.8.2 Contenido del archivo contenido.html en la implementación del *NokTemplate*.

Como se puede ver en el primer *template* se tienen 2 variables y en el segundo sólo una, para poder darle valor a ambas utilizarando PHP con un código como el que se muestra en la Tabla 1.8.3.

desarrollo.php

```
<?php
// Incluimos la Clase obviamente.
include ('Class.NokTemplate.php');
// Creamos una instancia del objeto.
// Definimos el lugar donde se encuentran los templates, en este caso './templates'.
$html = new NokTemplate('./templates');
// Cargamos los templates necesarios y le asignamos una clave, o sea tCuerpo hace referencia a cuerpo.html y tContenido a contenido.html
$html->cargar('tCuerpo','cuerpo.html');
$html->cargar('tContenido','contenido.html');
// Asignamos a la variable TITULO un texto. Titulo puede estar definida en cualquiera de las Plantillas.
$html->asignar('TITULO','Ejemplo número 1');
// Hacemos lo propio con Nombre.
$html->asignar('NOMBRE','Nok');
// Ahora, expandimos el contenido del template tContenido con sus variables ya asignadas, es decir, al hacer esto todo el contenido de la plantilla cuerpo.html se "asignará" a la variable contenido, con la salvedad que las variables que estén definidas dentro de la plantilla se les asignará su valor. Osea NOMBRE = 'Nok'.
$html->expandir('CONTENIDO', 'tContenido');
//y en CONTENIDO se vuelca lo que recién expandimos.
$html->expandir('FINAL', 'tCuerpo');
// Y por ultimo imprimimos la variable que contiene todo ya procesado. En este caso se llama FINAL pero puede ser cualquiera.
$html->imprimir('FINAL');
?>
```

Tabla 1.8.3 Contenido del archivo desarrollo.php en la implementación de *NokTemplate.*

A final de cuentas, cuando se quiere ver el resultado en pantalla arrojará algo como lo mostrado en la Figura 1.8.4.

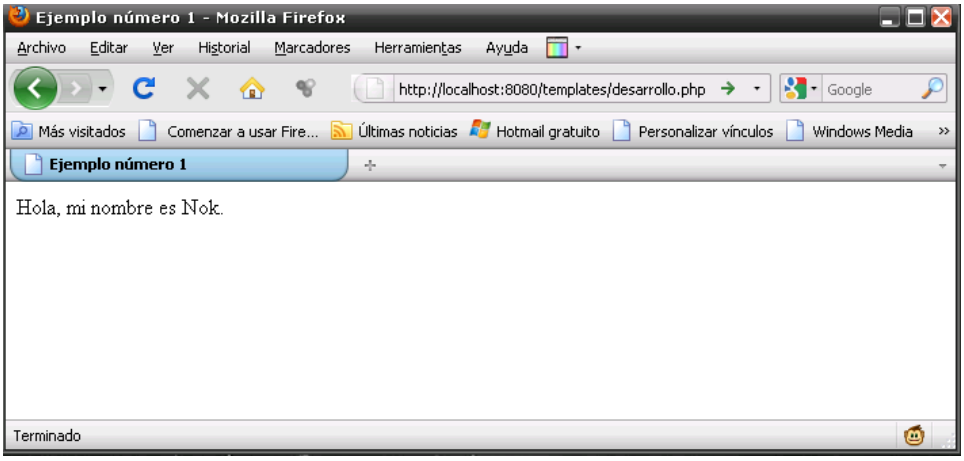


Figura 1.8.4 Salida de pantalla de la implementación e la clase *NokTemplate*.

Como se pudo observar es muy sencillo utilizar las plantillas o *templates*, ya que se puede hacer más independiente el código del diseño visual de la aplicación. Con esta clase también se pueden crear tablas, o incluso bloques, para facilitar la visualización de lo que se está haciendo, ya que al ser archivos separados no se pueden ver en un mismo archivo, es aquí donde entran los bloques para hacer precisamente esto. Un bloque se define dentro de la plantilla de esta manera:

```
<!—inicioBloque:NombreBloque -->
<!—finBloque:NombreBloque -->
```

Haciendo esto se podrá ver de una mejor manera el diseño de nuestro *template*.

1.8.3 Herramientas gráficas para PostgreSQL.

Las herramientas gráficas tienen como finalidad la de mejorar la visualización del contenido de las bases de datos, así como las consultas que se hagan a la misma, en general facilitan la administración de éstas.

Pg Access.

Como todos los gestionadores de bases de datos *PosgreSQL* también cuenta con un programa que permite darle un manejo gráfico a las consultas y lo que se trabaja en la base de datos, este es el caso de *pgaccess*, el cual una herramienta libre multiplataforma.

Esta herramienta permite realizar muchas funciones como crear, modificar y alimentar tablas, realizar consultas (con comandos o de manera gráfica), así como un diseñador visual para realizar formularios e informes.

La ventaja de utilizar este tipo de herramientas es que facilita el consultar la base de datos, además de que permite visualizar todo lo que se trabaje sobre ellas, su uso es totalmente recomendable para la administración, creación e implementación de aplicaciones que utilicen datos.

1.8.4 Funciones de PostrgreSQL para PHP.

Al igual que con otros manejadores de bases de datos, *PosgreSQL* también puede ser manipulado con PHP, haciendo la alimentación de la misma más dinámica, y existen funciones del lenguaje de programación que nos permiten interactuar con la base de datos, las más comunes y utilizables son:

- **pg_Connect(*host, puerto, opcuiones, tty ,basedatos*).** Abre una conexión con *PosgreSQL*.
- **pg_Close(*conexión*).** Cierra una conexión.
- **pg_affected_rows(*resultado*).** Regresa el número de campos afectados durante la ejecución de una sentencia *SELECT*, *UPDATE* o *DELETE*.
- **pg_delete(*conexión, nombre_tabla, arreglo_opciones*).** Borra los registros de la tabla indicada.
- **pg_insert(*conexión, nombre_tabla, arreglo_opciones*).** Inserta los campos con los valores definidos en *arreglo_opciones*, en la tabla *tabla_nombre*.
- **pg_num_rows(*resultado*).** Regresa el número de filas que contiene una consulta.
- **pg_num_fields(*resultado*).** Regresa el número de campos de una consulta.
- **pg_query(*conexión, consulta*).** Ejecuta una consulta.
- **pg_result_error(*resultado*).** Regresa un error de ejecución el la sentencia.
- **pg_select(*conexión, nombre_tabla, arreglo_opciones*).** Selecciona los campos especificados en el *arreglo_opciones*, de la tabla *nombre_tabla*.

- **pg_update(*conexión, nombre_tabla, arreglo_opciones*).** Modifica los campos de los valores establecidos en el *arreglo_opciones* de la tabla *nombre_tabla*.

Es así como se puede tener la idea de cómo se maneja PostgreSQL y su interacción con el lenguaje de programación descrito, con estas herramientas se pueden desarrollar cosas útiles y de gran calidad.

En conclusión, el diplomado me dejó una sensación de satisfacción personal, ya que reafirmé conocimientos que estaban un poco confusos y además aprendí nuevas formas de hacer las cosas con una mejor organización, calidad y buenos resultados. Existieron módulos donde faltó ahondar un poco más, sin embargo lograron el objetivo para poder desarrollar el proyecto final.

En los siguientes capítulos describiré lo visto en el los cursos de programación en JAVA, la importancia y aplicación de esta tecnología dentro de proyectos Web en la actualidad.

2. Programación JAVA.

Estos dos módulos de programación con JAVA tienen la finalidad de introducirnos en el lenguaje de programación, sus alcances, ventajas, desventajas e implementación en sistemas.

Java es un lenguaje de programación orientado a objetos desarrollado por la empresa Sun Microsystems en la década de los 90's. Su sintaxis se basa en los lenguajes C++ y C; adicionando un modelo de objetos más simple.

Características del lenguaje.

Una de las principales características de java es que sigue la metodología de programación orientada a objetos; la cual habla acerca de separar por entidades llamadas **objetos**, los cuales constan de ciertas características que los hacen únicos, **atributos** y a su vez pueden ejecutar ciertas acciones **métodos.**

Otra característica que lo hace resaltar de los demás es que debe ser ejecutado en cualquier sistema operativo, ya que hace uso de una JAVA VIRTUAL MACHINE (máquina virtual de java).

A la máquina virtual se le suma el uso del GARBAGE COLLECTOR (colector de basura), el cual funciona eliminando los elementos de memoria cuando ya no son usados por el flujo.

Tipos de programas en Java.

El lenguaje permite realizar diversos tipos de programas con características definidas para lograr hacer las tareas que se proponga, algunos ejemplos de éstos son:

- **Applets**. Son pequeños programas que se incorporan en una página Web compatible con Java para poder ejecutarse. A menudo se descargan junto con una página HTML desde un Servidor Web y se ejecutan en la máquina cliente.

- **Aplicaciones.** Son programas de propósito general que normalmente se ejecutan desde la línea de comandos del sistema operativo. Con Java se puede realizar cualquier programa que normalmente se crearía con algún otro lenguaje de programación.

- **Servlets.** Al contrario de los *applets* son programas que están pensados para trabajar en el lado del servidor y desarrollar aplicaciones Web que interactúen con los clientes y son una alternativa de la programación CGI tradicional.

2.1.1 Compilación y ejecución de programas.

En java existen diversas maneras de compilar y ejecutar programas, una vez instalado, en línea de comando se escribe: `javac archivo.java`.

Esto con la finalidad de interpretar el código, si todo está correcto, se genera un archivo **.***class* el cual se ejecuta de esta manera: `java nombreArchivo`.

También puede compilarse un programa en java con cualquier aplicación gráfica. Para usar esta opción y la anterior es forzosamente necesario haber instalado previamente java y tenerlo configurado en las variables de ambiente.

2.1.2 Creación de aplicaciones con el *JDK*.

Para instalar el *JDK* primero se tendrá que descargarlo de Internet directamente de http://java.sun.com/javase/downloads/index.jsp. Una vez instalado, hay que agregar estas líneas a las variables de ambiente:

```
SET PATH= [ruta]jdk1.2.2\bin;%PATH%
SET CLASSPATH=[ruta]jdk1.2.2\jre\lib;.
```

La variable de ambiente *CLASSPATH* le dice a la máquina virtual de java y otras aplicaciones en donde buscar las clases que se necesitan para ejecutar un programa. Una vez hecho estos cambios, se tiene que actualizar el *classpath* para indicarle al sistema operativo donde esta instalado java. Todo programa hecho con java debe de contar con una clase principal *main*, y será llamada primero que cualquier otra, y ésta a su vez llamará a los siguientes métodos de los cuales se conformará el programa, su sintaxis es la siguiente:

```
class miClase {
      public static void main(String args[]){
            System.out.println(“hola mundo”);
      }
}
```

2.1.3 Estructura del lenguaje.

La sintaxis del lenguaje java es muy simple, y se puede separar de la siguiente manera:

- *Comentarios*: hay de dos tipos, de una sola línea // y de varios párrafos /* */.

  También se pueden encontrar comentarios para documentación, los cuales son de la siguiente manera:

  ```
  /**@author: SoftTech*/
  ```

  De esta manera es más fácil obtener la documentación de los programas, con el comando `javadoc`.

- *Identificadores*: son aquellas variables que pueden generarse por el usuario, no pueden usar caracteres especiales.

- *Palabras clave*: Son palabras reservadas las cuales no pueden ser usadas como nombres de variables.

- *Expresiones y operadores*: una expresión es una combinación de variables, operadores y llamadas de métodos construida de acuerdo a la sintaxis del lenguaje que devuelve un valor. El tipo de dato del valor regresado por una expresión depende de los elementos usados en la expresión. Los operadores son símbolos

especiales que por lo común se utilizan en expresiones. Dentro de los más conocidos están:

- Operadores aritméticos (suma, resta, multiplicación, división y módulo).
- Operadores de asignación (Asignación, suma y asignación, resta y asignación, etc.).
- Operadores relacionales (igual, diferente de, mayor, que, menor que, etc.)
- Operadores especiales, incremento(a++ ó ++a), decremento (a-- ó --a), concatenación, etc.

Tipos de datos en java.

Para declarar una variable en java, debe especificarse el tipo de variable indicándolo junto con su identificador, la sintaxis es de la siguiente manera:

```
TipoSimple Identificador1, Identificador2;
```

Los tipos de datos pueden dividirse en dos categorías: simples y compuestos. Los primeros son tipos nucleares que no se derivan de otros tipos, como los enteros, de coma flotante, booleanos y de carácter; y Los compuestos se basan en los tipos simples, e incluyen las cadenas, las matrices y tanto las clases como las interfaces, en general.

2.1.4 Trabajando con clases y objetos.

La programación en java se basa en la programación de clases y objetos (POO), la cual se basa de dividir un programa como un conjunto de múltiples objetos para realizar alguna tarea en específico. Los objetos comparten un estado (definido por una o más variables), comportamiento (implementación de sus métodos) e identidad; contienen toda la información de que permite definirlo e identificarlo mediante otros objetos que pertenecen a otras clases.
Los objetos están compuestos de dos partes:

- **Atributos:** es una característica que define a mi objeto de los demás, en java los atributos son las variables de mi clase.

- **Métodos:** son las acciones que mi objeto puede realizar. Para java un objeto es representado por una clase, la cual se define de la siguiente forma:

```
a) public class miClase(){
b) string variable;
c) int variableInt;
d) public static void main(String[] args){}
e) }
```

En la línea **a** se tiene la definición de mi clase, las líneas **b** y **c**, son los atributos de mi clase, y la línea **d** define un método de mi clase aunque también puede llamarse función.

Todo programa en java debe contener al menos una clase *main* (línea **d**), ya que para el interprete de java será la encargada de llamar a sus demás métodos del programa.

2.1.5 Arreglos.

Un arreglo es una colección de objetos numerados del mismo tipo, en donde cada variable o celda en el arreglo tiene un índice. Las celdas están numeradas del 0 al N-1, donde N es el número de celdas del arreglo es decir su capacidad.

Los índices de un arreglo en Java deben estar dentro de los límites, 0 – N-1, de lo contrario se generará un error durante la ejecución.

Para utilizar un arreglo en Java hay que:

1. Declarar una variable para que contenga el arreglo.
2. Crear un nuevo arreglo de objeto y asignarlo a la variable de arreglo.
3. Almacenar información en el arreglo.
4. Realizar operaciones de almacenamiento y recuperación con los elementos del arreglo.

Para declarar un arreglo se usan **[],** por ejemplo

```
String arregloUno [ ];
 int arregloDos [ ];
```

Después de declarar nuestro arreglo hay que inicializarlo, es decir, decirle a la máquina virtual de java el tamaño que va a tener:

```
String [ ] nombreJugadores = new String [10];
        int [ ] temps = new int [99];
String [ ] chiles = { “jalapeno”, “de árbol”, “serrano”, “habanero”};
```

También pueden declararse arreglos bidimensionales de la siguiente forma:

```
          final int MAXT = 11;
int [] [] tabla = new int [MAXT][MAXT];
```

2.1.6 Herencia.

La herencia en java es un tema realmente importante, ya que con esta característica se puede reutilizar el código sin necesidad de ser reescrito, gracias a esta característica se pueden construir nuevas clases partiendo de una jerarquía existente.

La herencia facilita la creación de objetos a partir de otros ya existentes, obteniendo características (métodos y atributos) predefinidas.

La herencia parte de una clase principal, sus métodos y atributos son compartidos por las clases que la preceden; las clases principales son llamadas padres y las que de ellas se derivan, hijas.

En java, la herencia se define por la palabra reservada **extends,** como se podrá observar a continuación:

```
class MiClase extends TuClase(){ }
```

Con esto se podría acceder a los atributos y métodos de la clase TuClase a través de la clase MiClase.

2.1.7 Polimorfismo.

El concepto de polimorfismo es fundamental para java, etimológicamente se conforma de dos partes: *poli* = múltiple y *morfismo*= formas, esto significa que un mismo Objeto puede tomar diversas formas, permitiendo que al momento de programar un objeto éste pueda tener una forma genérica que haga más flexibles las llamadas a sus métodos.

El polimorfismo se presenta cuando se tiene una interfaz definida normalmente en una clase abstracta, la cual compartirán las implementaciones de esta clase, teniendo así una referencia de un tipo abstracto, por medio de dicha interfaz, puedes almacenar y hacer referencia a una instancia de alguna de las diferentes implementaciones del tipo abstracto, haciendo al código más acoplable, lo cual dará mucha flexibilidad permitiendo su mejor mantenimiento.

2.1.8 Excepciones.

Las excepciones en Java están destinadas a la detección y corrección de errores en tiempo real. Si hay un error, la aplicación no debería detenerse y generar un error. Java sigue el mismo modelo de excepciones que se utiliza en C++. Utilizadas en forma adecuada, las excepciones aumentan en gran medida la robustez de las aplicaciones.

En Java cuando ocurre un error dentro de un método, crea un objeto *Exception*, el cual contiene información sobre la excepción, que incluye su tipo y el estado del programa cuando ocurrió el error. El sistema de ejecución es el responsable de buscar algún código para manejar el error. El manejo de excepciones en Java sigue una estructura como la siguiente:

```
try {
   //Codigo donde puede ocurrir un error
}
catch (ExcepcionA ex) { // Que se va a hacer en caso que se lance
una Excepcion A
}
...
catch (ExcepcionZ ex) { // Que se va a hacer en caso que se lance
una Excepcion Z
}
```

Dentro del bloque *try{ }* viene encerrado la parte del programa que se desea manejar sus excepciones. El código dentro de algún *catch* (*TipoExcepcion e*) se ejecuta en caso de que lance una excepción *TipoExcepcion* o que pertenezca a este grupo. El sistema de ejecución Java busca hacia atrás en la pila de llamadas para encontrar el método que esté interesado en manejar una excepción particular.

Sentencia THROW.

Todos los métodos Java utilizan la sentencia *throw* para lanzar una excepción. Esta sentencia requiere un sólo argumento, un objeto **Throwable**.

```
public Object pop() throws EmptyStackException {
   Object obj;
   if (size == 0)
      throw new EmptyStackException();
   obj = objectAt(size - 1);
   setObjectAt(size - 1, null);
   size--;
   return obj; }
```

2.2 Desarrollo de aplicaciones web con JAVA, JSPs y SERVLETs.

En el primer curso se vieron los aspectos básicos y fundamentales para trabajar con JAVA, ahora se describirá la manera de implementarlo en una aplicación web para sacarle mayor provecho a las bondades que brinda a los programadores.

La construcción de sistemas en capas ha demostrado ser una buena práctica para desarrollar sistemas, ya que hace de esta tarea algo más organizada, ya que el objetivo principal consiste en separar las partes del sistema, además de tener grandes ventajas como:

- Fáciles y rápidos de mantener.
- Fáciles de entender.
- Fácil integración con otros sistemas.
- Más seguros.

Capa web.

Es también conocida como capa de presentación, aquí se encuentran todos los programas que ayudan a presentar los datos ya procesados por otras capas al usuario final, por lo que su objetivo principal es el de lograr la correcta comunicación con éste. Entre los más conocidos se encuentra *Struts* y *Java Server Faces.*

Capa lógica de negocios.

Aquí se encuentran los componentes que manejan la lógica de negocios del sistema. Su funcionamiento es sencillo, recibe las peticiones de la capa de presentación, procesa la lógica de las mismas y deja los datos correctamente procesados para que la capa de presentación los utilice.

Capa de Persistencia.

Se encarga de procesar de una manera sencilla y eficiente las conexiones con algún RDBMS en particular. Por lo general se utiliza un *framework* para su implementación como *Hibernate*.

Capa de Integración.

Es la capa intermedia entre la lógica de negocio y la de persistencia. Encapsula la lógica para interactuar con la capa de persistencia.

2.2.1 Configuración del servidor web.

También conocido como contenedor, el servidor web es parte del *software* que tiene un trabajo muy importante en el desarrollo de aplicaciones, se pueden definir con los siguientes conceptos:

- **Comunicación**. Provee de un mecanismo de comunicación que permite interactuar con los *servles*.
- **Manejo automático de ciclo de vida**. controla el inicio y la muerte de los *servlets* del sistema.
- **Multihilo**. Automáticamente crea un nuevo hilo para cada petición que recibe.
- **Configuración declarativa**. La configuración de aplicaciones web, requiere de un archivo *XML*, por lo que ya no es necesario volver a compilar el código.

Instalación del Servidor Glassfish.

En el curso se instalo este servidor web para contener proyectos web, fue desarrollado por *Sun Microsystems* e implementa tecnología *J2EE* además de ser de código abierto, es muy utilizado y además presenta una gran compatibilidad con la *IDE* de *Eclipse*.

Su instalación y configuración se logra siguiendo los pasos que a continuación se desglosan:

- Descargar el archivo *.jar* de la página oficial de *Glassfish*.
- Crear la carpeta *C:\Servers*.
- Copiar el archivo en la carpeta que se acaba de crear.
- En línea de comandos, posiciona en la ruta de la carpeta anterior y ejecuta la siguiente línea:

```
java –Xmx256m –jar nombreFichero.jar
```

- Cambiar a la carpeta que se acaba de crear *C:\Servers\glassfish* y crear el *home* de *JAVA* de la siguiente manera:

```
set JAVA_HOME=C:\ruta del fichero jdk16.0.0.7
```

- Ahora se ejecuta: `lib\ant\bin\ant –f setup.xml`.
- Cambiar al directorio C:\Servers\glassfich\bin y se ejecuta: `asadmin start –domain domain1`. Una vez que se ejecuta mostrará los puertos por los que escuchará el servidor de aplicaciones.
- En un navegador ejecutar `http://localhost:4848`, pedirá un nombre de usuario y una contraseña, el primero es `admin` y el segundo `adminadmin`.

Con esto se tiene ya trabajando el servidor de aplicaciones y desde la dirección que ingresa al final de la configuración se podrá entrar a todos los elementos de *Glassfish*, dentro de las acciones más importantes está la de los permisos de ejecución de los proyectos web, es importante deplorar la aplicación para poder verla en acción

Como se mencionó en un principio se utiliza la *IDE* de *Eclipse* como editor de código, entre otras cosas, para los proyectos, así que una parte importante es la de integrar el servidor web con la *IDE* que se esté utilizando para poder hacer más manejable la publicación de los que se desarrollan, este proceso es sencillo sólo hay que seguir los siguientes pasos:

- Desde *Eclipse* se va al menú *Window > Show View > Server*.

- Posicionar en la parte donde muestra la compilación Eclipse, con el botón secundario del mouse desplegar las opciones y seleccionar *New > Server*.

- Aparecerá una ventana en donde se selecciona la opción *Download additional server adapter*, y el asistente realizará una búsqueda de los servidores de aplicaciones disponibles, en cuanto termine se selecciona el que interesa que es *Glassfish*.

- Se reiniciará la *IDE* ya con el servidor integrado, ahora si se puede comenzar el desarrollo de aplicaciones.

2.2.2 Introducción a los SERVLETs.

Los *servlets* son un mecanismo con el que se pueden construir aplicaciones web, ya que llevan el lenguaje *JAVA* a otro nivel, permitiendo desarrollar proyectos de gran complejidad de una manera eficaz, eficiente y relativamente sencilla.

Definición.

El propósito fundamental de estos elementos es la de procesar las peticiones de los clientes y enviar una respuesta. Las peticiones contienen información importante que el *servlet* debe ser capaz de encontrarla y saber como utilizarla, así mismo la respuesta que envía al navegador debe saber como mandarla para que éste pueda publicarla de manera correcta.

Ciclo de vida del servlet.

Esto se refiere a los diversos estados en los que se puede encontrar un *servlet*. El ciclo de vida de estos elementos se puede determinar de la siguiente manera:

- El cliente hace una petición y la envía al servidor.

- El servidor de aplicaciones detecta que la petición es para un determinado *servlet* y crea dos objetos: *HttpServletRequest* y *HttpServletResponse*.

- El contenedor encuentra el *servlet* requerido para procesar la respuesta y crea un nuevo hilo para esa petición y llama al método ***service()*** y le pasa los objetos *request* y *response* como parámetros.

- El servidor web invoca al método ***service()*** y dependiendo de la petición del cliente, se genera cualquiera de estos dos métodos: ***doGet()***, si la petición la hizo por el *GET*; o ***doPost()***, si la realizo por este método.

- El método requerido, cualquiera de los dos mencionados anteriormente, realiza su trabajo, elabora una página dinámicamente, de así requerirlo y se agrega todo lo que será enviado al cliente.

- Cuando el método ***service()*** termina su tarea, el hilo que generó se muere, los objetos *request* y *response* también desaparecen y finalmente el cliente obtiene una respuesta.

Método init().

El servidor es el encargado de llamar a este método, y lo hace después de que el *servlet* es creado pero antes de que genere una respuesta. Su objetivo es el de inicializar el *servlet* antes de procesar la información.

Método service().

Es invocado cuando se recibe una petición generando un nuevo hilo para que ésta sea atendida. Su tarea principal es la de determinar el método *HTTP* en el cual fue hecha la petición (*GET*, *POST*, etc), para poder invocar al método correcto en el *servlet*, ya sea *doGet()*, *doPost()*, etc.

Método destroy().

Es invocado por el servidor web antes de quitar la instancia del *servlet*, esto sucede cuando el contenedor es dado de baja y necesita liberar memoria, o cuando un *servlet* pasa determinado lapso de tiempo sin recibir peticiones. Por lo que este método se invoca cuando todos los *servlets* han terminado su trabajo, liberando los recursos que éste ocupaba.

Método doGet() y método doPost().

El método *service()* es el encargado de invocar a estos métodos, dependiendo del tipo de petición *HTTP* que haya obtenido. Si la petición del cliente fue enviada por el *GET*, el método *service()* invocará a *doGet()*; y si fue por el *POST*, *service()* invocará a *doPost()*.

Es en estos dos métodos donde se encuentra la lógica de negocios de las aplicaciones, es decir, aquí se encuentra la implementación del código para que los proyectos ejecuten las tareas para las que fueron creados.

Métodos de la interfaz HttpServletRequest.

Esta interfaz es importante dentro de la estructura del proyecto ya que va a ser la encargada de proveer de los métodos útiles para poder establecer una comunicación con el cliente,

mediante el protocolo *HTTP*, ya que nos permitirá trabajar con *cookies*, cabeceras y sesiones.

Los métodos más utilizados por la interfaz *HttpServletRequest* se encuentran representados en la Tabla 2.2.1.

<<interface>> **HttpServletRequest**
getContextPath() getCookies() getHeader(String) getSession() getMethod()

Tabla 2.2.1 Diagrama *UML* de la clase *HttpServletRequest*.

Métodos de la interfaz HttpServletResponse.

Los métodos de esta interfaz tienen el propósito de trabajar con el protocolo HTTP y manejar las principales cuestiones de la respuesta enviada al cliente como errores, *cookies* y cabeceras.

Los métodos más utilizados por la interfaz *HttpServletRsponse* se encuentran representados en Tabla 2.2.2.

<<interface>> **HttpServletResponse**
addCookie() addHeader() encodeRedirect() sendError() sendStatus()

Tabla 2.2.1 Diagrama *UML* de la clase *HttpServletResponse*.

2.2.3 Elementos del JSP.

El servidor de páginas Java o *JSP* por sus siglas en inglés (*Java Server Page*) es una herramienta muy poderosa para ejecutar la presentación de la conformación al usuario. En una aplicación web los *servlets* se encargan de la lógica de negocios y las *JSPs* de acomodar de cierta manera la información para presentarla al usuario final.

Un *JSP* es un documento que contiene código java y etiquetas *HTML*, lo interesante es que al final de cuentas éstos también son procesados como un *servlet* por el servidor web ya que los compila como tales, los instancia, inicializa, genera el hilo que atenderá la petición e invoca al método *service()*.

Ciclo de vida de las JSPs.

Como se explicó anteriormente las *JSPs* son convertidas a *servlets*, lo importante es saber como lo hace el servidor web, he aquí el procedimiento:

- Se crea el *JSP*, se almacena en el contenedor y se lleva a cabo la acción *deploy* en la aplicación web. El servidor de web lee el archivo *web.xml* de la aplicación pero no le hace nada a los archivos *.jsp*.

- Cuando el cliente solicita un archivo *.jsp*, el contenedor lo encuentra y trata de convertirlo a un *servlet*, es decir en un archivo *.java*.

- El servidor web trata de compilar el archivo *.java* y si no contiene ningún error genera el archivo *.class* correspondiente.

- El contenedor carga el archivo *.class* en memoria.

- El servidor de aplicaciones instancia y crea el objeto, es decir, invoca a su constructor y al método *jspInit()*, en este punto el objetos es un *servlet* y esta listo para aceptar peticiones.

- El contenedor creo un hilo para poder atender la petición que recibió, por lo tanto el método *jspService()* es invocado.

Sintaxis y semántica de JSP.

Dentro de un JPS se pueden encontrar diversos mecanismos que contienen código java, aunque no es lo más recomendable, dado que lo hacen más difícil de depurar y mantener, esto se ha convertido en una práctica cotidiana, por lo que se deben conocer todos los elementos que puede contener este tipo de archivos.

Expresiones y scriptlets.

Un scriptlet es identificado porque se escriben entre las etiquetas **<% %>**, donde se puede encontrar código java común. Los scriptlets le dicen al servidor de aplicaciones que con base en la fase de traducción, no realice más trabajo, pues ya es código java.

Una expresión es identificada por estar entre las etiquetas **<%= %>** y tienen la función de imprimir todo el código que se encuentra entre éstas. Las expresiones no evitan estar escribiendo varios *out.print()*.

Declaraciones.

Una declaración se puede identificar porque se escriben entre las etiquetas **<%! %>**, estas crean miembros de la clase (métodos y atributos) lo que genera que todo lo que se encuentra entre estas etiquetas queda fuera del método *service()* para el *servlet* que crea el contenedor.

Directivas de página.

Estas directivas son una manera de darle instrucciones al contenedor al momento de estar procesando un *JSP*, se distinguen por escribirse entre las etiquetas **<%@page %>.**

Existen 13 atributos que se pueden incluir en esta directiva, aunque la mayoría se utiliza raramente, algunos de los atributos que se pueden incluir definen cosas como el tipo de contenido que la página enviará en la respuesta, especificar si la página tendrá una sesión implícita o definir si se trata de una página de error.

Objetos implícitos en las JSPs.

Cuando el contenedor convierte al *JSP* en *servlet*, existen varias declaraciones y asignaciones implícitas, teniendo esto en cuenta se puede construir un JSP dando por hecho que lo que escribamos será parte de un *servlet*, con la idea de que el contenedor es el encargado de definirlos de manera correcta. Algunas de estas declaraciones son:

- Out.
- Request.
- Response.
- Session.
- Application.
- Config.
- Exception.
- pageContext.
- Page.

2.2.4 Formularios HTML.

La importancia de los formularios es que le permiten al cliente tener comunicación con el servidor por medio de elementos gráficos que permiten ingresar información. Cuando un formulario o forma es enviada al servidor, toda la información que contiene es empaquetada y enviada a un servidor para ser procesada por una aplicación web.

Métodos GET y POST.

El método determina la manera en que la información ingresada en el formulario se enviará al servidor. Básicamente ambos métodos hacen la misma tarea, que es enviar información al servidor, la diferencia está como lo hace cada uno; mientras que el *POST* empaqueta todos los datos del formulario y los envía dentro del documento *HTTP*, el *GET* también empaqueta la información sólo que éste la añade al final de la *URL* antes de enviarla al servidor.

Para definir qué método es el que se van a utilizar y cuál es la información que se desea enviar al servidor utiliza la etiqueta *HTML* `<form></form>`. Dentro de éstas se escribirán e incluirán todos los elementos que se quieren procesar mediante cuadros de texto, cuadros de opción, casillas de verificación o cualquier otro elemento *HTML* que compone a un

formulario. También se podrá definir el método por el que se desean sean enviados los datos con el atributo *method*; así como especificarle al navegador donde tiene que enviar la información con el atributo *action*.

2.2.5 JAVABEANS.

Los *JavaBeans* son componentes del software java, son programas reusables que se utilizan para crear aplicaciones sofisticadas; además de que se pueden corregir, desarrollar y modificar fácil y rápidamente.

JavaBeans es una especificación para la creación de programas. Si se tiene el conocimiento de que una clase tiene este tipo de especificación cualquier persona o programa sabrá como utilizarla correctamente.

Propiedades de los beans.

Una clase que se apega a este tipo de especificación debe seguir los puntos que se establecen en la misma. Los atributos de la clase deben ser de acceso privado, para que la única manera de acceder a ellos sea por medio de los atributos públicos de la misma. Los métodos que cambian el valor de un atributo se llaman ***setter's***; y los que devuelven el valor de un atributo, ***getter's***. *JavaBeans* establece las reglas para nombrar a estos miembros de la clase siendo los más importantes:

- Si el atributo no es booleano, el nombre del método *getter* debe comenzar con el prefijo *get*.

- Si el atributo es booleano, el nombre del método *getter* debe comenzar con el prefijo *get* o *is*.
- El nombre del método setter debe comenzar con el prefijo set.

- Para completar el nombre del método *getter* y *setter*, la primera letra del atributo se debe de cambiar a mayúscula y posteriormente ponerla al final.

- El método *setter* debe tener un control de acceso público y un valor de retorno vació, así como recibir un único argumento, el tipo de dato de este argumento debe ser el mismo que implica el atributo.

- El método *getter* debe tener un control de acceso público, no recibir argumentos y el valor de retorno debe ser del mismo tipo de dato del atributo.

Etiqueta <jsp:useBean>.

Esta es la forma de decirle al contenedor que se utilizará un *bean* de una clase, ya que declara e inicializa el *bean* de un objeto que será utilizado por otras etiquetas posteriormente.

Etiqueta <jsp:setProperty>.

Es una acción estándar que sirve para invocar a los métodos *setters* de una clase que se apega a las especificaciones de *JavaBeans*.

Etiqueta <jsp:getProperty>.

Es una acción estándar que sirve para invocar a los métodos *getters* de una clase que se apega a las especificaciones de *JavaBeans*.

2.2.6 Sesiones.

El servidor de aplicaciones se olvida de un cliente tan pronto le envía una respuesta, es decir, los contenedores no recuerdan que peticiones han recibido de que clientes y tampoco recuerda que fue lo que les mando como respuesta. En ocasiones esto no tiene la mayor importancia, pero existen veces en la que es importante establecer una conversación con el servidor para recordar algunas peticiones y respuestas emitidas por el mismo hacia un cliente en especial.

La manera en que un contenedor recuerda a un cliente es la siguiente: el cliente hace una petición, el servidor de aplicaciones genera un identificador único para ese cliente y se lo envía a éste en la respuesta. El cliente tiene que enviar este identificador en las subsecuentes peticiones y el contenedor verifica si éste ya existía o no.

Creación de sesiones.

Una de las ventajas del manejo de sesiones es que el desarrollador no se tiene que preocupar por generar el identificador, ni tampoco ver si éste ya existía o no, lo único que tiene que hacer es especificarle al servidor web si quiere, iniciar o recuperar una sesión.
El servidor lo que hace es generar el identificador, crear el objeto de la clase *Cookie*, instanciarlo con el valor del identificador y anexarlo como parte de la respuesta que se enviará al cliente.

Guardar y recuperar objetos a partir de sesiones.

Cuando se crea una sesión se puede, guardar, recuperar o borrar cualquier objeto de la misma:

- **`void setAttribute(`*`java.lang.String nombre, java.lang.Object valor`*`)`**, guarda un objeto en la sesión actual.

- **`java.lang.Object getAttribute(`*`java.lang.String nombre`*`)`**, recupera un objeto asociado a una sesión a través del nombre del objeto.

- **`void removeAttribute(`*`java.lang.String nombre`*`)`**, borra un objeto asociado a la sesión.

Destrucción de sesiones.

Las sesiones ocupan recursos del servidor de aplicaciones, por lo que no es muy recomendable que pase esto durante un largo periodo de tiempo, y como el contenedor no sabe identificar cuando un cliente termino o se fue, lo mejor es destruir la sesión para que los recursos que ésta ocupaba se liberen, hay tres formas de destruir una sesión:

- Configurando un tiempo transcurrido en el *Deployment Descriptor*.
- Configurando el tiempo transcurrido de esta manera: **`session.setMacInactiveInterval(900);`**
- Invocando al método ***`session.invalidate().`***

2.2.7 Java Standard Tag Library (JSTL).

Son etiquetas personalizadas que sirven de medio para poder utilizar código Java en un JSP, las cuales ya estas preestablecidas, solo hay que implementarlas dependiendo nuestras necesidades.

JSTL define operaciones comunes dentro de un *JSP*, las etiquetas que ofrece están orientadas a eliminar el uso de los *scriptlets*.

Expression Languaje (EL).

Forma parte de la especificación *JPS 2.0* y se utiliza para llevar a cabo operaciones que se realizan con los *scriptlets* y las expresiones; así como para invocar de una manera más sencilla código Java. Este tipo de código java no se encuentra en el *JSP*, sino en otro lado lo que el *Expresión Languaje* hace es llamarlo desde el *JSP* de esta manera:

```
<html>
<body>
      Tu nombre es: ${persona.nombre}
</body>
</html>
```

Etiquetas.

Existen muchas etiquetas dentro del JSTL, por lo que no será muy necesario hacer muchas personalizadas, existen cuatro librerías bastante grandes:

- **Core**, tiene etiquetas que se pueden utilizar para propósito general.
- **Formatting**, etiquetas para dar formato a ciertos tipos de datos, como a una fecha.
- **XML**, etiquetas para el manejo de documentos de este tipo.
- **SQL**, librería para la interacción de bases de datos relacionales a través de secuencias *SQL*.

2.2.8 Acceso a bases de datos con JDBC.

Una de las partes fundamentales cuando se desarrollan aplicaciones web es la de la interacción con las bases de datos, en Java existe el **API JDBC** (*Java Data Base Conectivity*), una herramienta sencilla y poderosa para interaccionar con las bases de datos con código Java.

Acceso a bases de datos.

Utilizando el JDBC es muy sencillo interactuar con las bases de datos mediante sentencias estructuradas *SQL*, y este *API* presenta una gran ventaja, que si la aplicación cambia de **DBMS** (*Batabase Managment System*), los cambios en el código serían mínimos.

Establecer una conexión.

Como se describió anteriormente con el JDBC se puede manejar el acceso a bases de datos relacionales, éste ayuda a realizar tres tipos de tareas:

- Establecer una conexión con la base de datos.
- Ejecutar sentencias *SQL*.
- Procesar el resultado de las sentencias *SQL*.

Interacción con bases de datos en aplicaciones web.

Esta tarea es la fundamental cuando se habla de aplicaciones web, ya que mostrará lo que se tiene almacenado con una presentación llamativa, la implementación del API JDBC es sencilla, sólo se necesita descargar el conector o controlador del manejador de bases de datos que se van a utilizar, agregarlo al IDE de desarrollo y utilizarlo de la mejor manera.

Como se pudo observar este lenguaje permite realizar proyectos web robustos de una manera organizada debido a que es orientada a objetos, así mismo facilita la interacción con los manejadores de bases de datos más comunes con una compatibilidad completa. Por lo que sin lugar a dudas esta es una opción viable para implementarse.

Este par de cursos fueron de gran ayuda para comenzar a trabajar y conocer JAVA, aunque faltan de conocer varios aspectos para trabajarlo de manera mas intuitiva, sin embrago me pareció correcto el manejo de los temas y la forma en que se llevaron a cabo.

En el próximo capítulo se expondrá el proyecto final presentado en el Diplomado de Diseño de sistemas con Software Libre, el cual consta de la captura y turno de quejas y reclamaciones.

3. Proyecto de un Sistema de Registro de Quejas

La culminación del diplomado se da con la presentación de un proyecto que conjugue todo lo que se vio a lo largo de sus ocho módulos, desde la planeación administración, diseño, programación, conexión a bases de datos y seguridad, por lo que en el presente capítulo se describirá de manera sistemática como fue que se desarrollo el sistema de registro de quejas.

El CONSEJO NACIONAL PARA PREVENIR LA DISCRIMINACIÓN, *CONAPRED*, es un órgano de Estado creado por la Ley Federal para Prevenir y Eliminar la Discriminación, aprobada el 29 de abril de 2003, y publicada en el Diario Oficial de la Federación (*DOF*) el 11 de Junio del mismo año. El Consejo es la institución rectora para promover políticas y medidas tendientes a contribuir al desarrollo cultural y social y avanzar en la inclusión social y garantizar el derecho a la igualdad, que es el primero de los derechos fundamentales en la Constitución Federal.

El *CONAPRED* tiene como una de sus principales tareas poner a disposición de las personas los medios para defender su derecho a no ser discriminadas. Para ello, toda persona que considere que ha sido objeto de un acto discriminatorio puede acudir a la Dirección General Adjunta de Quejas y Reclamaciones para denunciarlo.

Se puede poner una queja si el presunto responsable de la conducta discriminatoria es un particular, o una reclamación si se trata de un servidor público federal.

La denuncia deberá presentarse por escrito con los datos y la firma del agraviado (persona o grupo de personas). En caso de hacerla por teléfono o correo electrónico, deberá ser ratificada por escrito durante los cinco días hábiles siguientes, a menos que se esté fuera del D. F. o privado de la libertad.

Con objeto de eliminar la conducta discriminatoria, el *CONAPRED* iniciará un proceso conciliatorio entre el agraviado y el presunto responsable. De no lograrlo, el *CONAPRED* orientará a la parte agraviada sobre las alternativas correspondientes.

Para el *CONAPRED* es vital el contacto con la sociedad, ya que sólo así podrá luchar efectivamente contra la discriminación. Por ello es muy importante denunciar cualquier acto discriminatorio. Recordar que el artículo primero de la Constitución prohíbe toda forma de discriminación.

3.1 Objetivo.

Diseñar un sistema con software libre en Intranet dirigido al personal adscrito a la Dirección General de Quejas y Reclamaciones, en específico a la Jefatura de Recepción, Registro y Turno del *CONAPRED*, que le permita capturar los datos personales mínimos para presentar una queja o reclamación en esta institución, optimizando la consulta y registro de las mismas.

3.2 Descripción.

En la Jefatura de Recepción, Registro y Turno, en lo subsiguiente *JRRT*, se realizan diversas actividades que pueden optimizarse, por lo que el Sistema de Captura y Consulta de Denuncias (*SICACODE*) le facilitará las siguientes tareas:

- **Registro de Quejas y Reclamaciones.** Este proceso se optimizará creando formularios con los campos correspondientes a las personas legales (peticionario, agraviado, presunto discriminador) que intervienen en este tipo de denuncias, así como la captación de los documentos que se presentan y antecedentes registrados del peticionario.

- **Búsqueda.** Es un formulario que tendrá filtros de búsqueda tales como peticionario, agraviado, presunto discriminador y fecha de registro, las cuales se realizarán mediante consultas a la base de Datos.

- **Generación de Reportes.** Una de las funciones de la *JRRT*, es la de reportar diaria, semanal, mensual y anualmente el total de ingresos tanto de quejas como de reclamaciones, separadas por medio de recepción y por presunto tipo de discriminación.

- **Estadísticas.** Es otra de las tareas de la *JRRT*, ya que es el área a la que llegan las solicitudes de información del *IFAI*, es por esto que se requiere de un módulo que contenga gráficas de la información básica. Las gráficas a desarrollar serían: total de quejas y reclamaciones por, género, estado y total de las mismas.

3.3 Administración del servidor.

Lo que se pretende para este sistema es tener un servidor dedicado, ya que la información que se manejará es altamente confidencial y no se pueden correr riesgos en la seguridad de los datos, es por eso que el manejo de los mismos se debe realizar con cuidado; para esto se particionará el disco duro de la siguiente manera, indicando también del tipo de sistema de archivos que tendrán:

- **/dev/sda1** (swap) de 1Gb.
- **/dev/sda2** (reiserfs) de 50Gb.
- **/dev/sda3** montado permanentemente en **/mnt/base** (reiserfs) de 100 Gb.
- **/dev/sda4** montado permanentemente en /**mnt/respaldo** (reiserfs) de 200 Gb.

Debido a la independencia del *SICACODE*, los servicios que deberán permanecer activos en el servidor, son los básicos para que se pueda contar tanto con un buen funcionamiento como administración quedando éstos de la siguiente forma:

- **Shh,** para poder monitorear de manera remota el funcionamiento del mismo, así como para darle mantenimiento preventivo de software, ya que para el hardware se tendrá que hacer en sitio.

- **Servidor web,** el que se va a ocupar va a ser Apache, ya que permite una mejor interacción con php y será vital para el funcionamiento del sistema.

- **PHP,** lenguaje de programación utilizado para la construcción del sistema, por lo que deberá permanecer activo.

- **MySQL,** necesario para la interacción del sistema con la Base de datos del mismo, fundamental para el la consulta de información y móvil principal del sistema, por lo que debe estar activado siempre.

3.4 Usuarios y reglas de buen uso del servidor.

Como servidor dedicado, contará dos usuarios básicos: *root* y *mysql*, ya que éste no brindará servicio de ningún tipo a los usuarios del sistema, sólo lo alojará. Los propósitos de estas cuentas son los siguientes:

- *Root* – será, como en todos los Linux, el superusuario, necesario para poder realizar todas las tareas de mantenimiento preventivo y correctivo del servidor, así como también permitirá llevar acabo la administración del mismo.

- *Mysql* – éste será el encargado de levantar el servicio correspondiente a la base de datos del *SICACODE*, para así poder tener una mejor ubicación de los procesos a la hora de detectar alguna falla.

En lo que se refiere a las reglas de buen uso del servidor, cabe destacar que solo son aplicables al administrador del mismo, ya que no cuenta con servicios adicionales para los usuarios. Estas reglas son las básicas para cualquier administrador:

1. Asegurara un buen funcionamiento del servidor como de la aplicación y sus servicios.

2. Mantener confidencialmente los datos que se encuentran en la Base de Datos, ya que éstos son totalmente reservados, debido al carácter de los asuntos atendidos por la institución en la que se va a desarrollar el sistema.
3. Verificar la integridad de la información en los respaldos así como mantenerlos en un lugar seguro para su custodia.

4. Establecer vías de comunicación efectivas con los usuarios del *SICACODE*, para prevenir posibles fallas de los servicios, el sistema o el servidor.

3.5 Respaldos.

Para un mejor control y seguridad de los datos se contará con tareas programadas para realizar respaldos periódicamente de la información más importante, principalmente la base de datos y de manera más temporal los scripts que contiene el sistema; quedando éstas de la siguiente manera:

- *Bases de datos*, se respaldará automáticamente cada domingo a las 9 de la mañana, para así evitar pérdidas de información y no obstaculizar las tareas normales de los

usuarios. Este respaldo será guardado con el nombre "*BKUPDB_[fecha de creación].tar*", y será depositado en la partición destinada para este fin.

- *Scripts,* Serán respaldados cada mes, esto debido a que se estima que no sufran modificaciones importantes en un periodo corto de tiempo, cabe destacar que se contará con un respaldo inicia de los mismos. Los respaldos de esta información serán almacenados en la partición especial para este fin *(/mnt/respaldos)*, y se le darán el nombre de "*BKUPSC_[fecha de creación].tar*".

Aparte de contar con respaldos en la partición */mnt/respaldos*, también se crearán 2 copias en DVD's con la misma información, una de las cuales se le quedará a la institución y la otra será guardada por el administrador del sistema.

3.6 Diseño de pantallas.

El diseño se hizo con HTML y las pantallas representativas del sistema están estructuradas como se enuncian a continuación.

3.6.1 Ingreso al sistema.

La Figura 3.6.1 muestra la pantalla de ingreso al sistema, debe de introducir su usuarios y su contraseña y presionar *Entrar*, si son correctos ambos accederá a la pantalla de registro.

Figura 3.6.1 Pantalla de ingreso al sistema

Una vez validado el usuario y la contraseña ingresada se accede a la *pantalla principal* del sistema, la cual, esta dividida en 6 área (Figura 3.6.2).

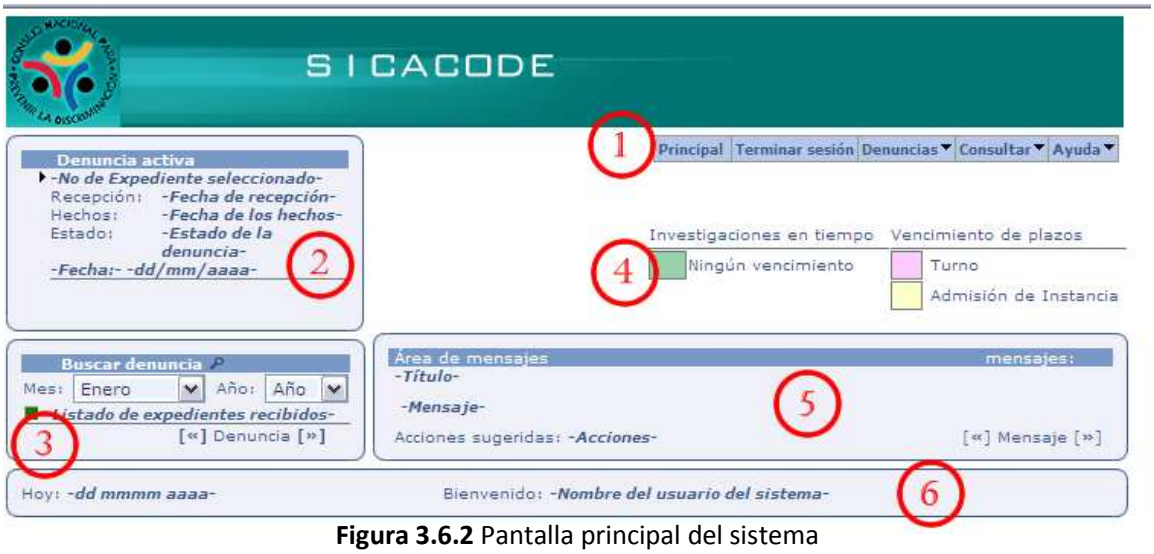


Figura 3.6.2 Pantalla principal del sistema

1) *Menú Principal*, contiene las opciones para ingresar a las diferentes pantallas, bastara con posicionarse en la opción deseada para que se desplieguen los submenús que contenga el elemento seleccionado.

2) *Denuncia activa*, muestra la queja o reclamación con la cual se está trabajando actualmente, y las acciones que se seleccionen en el *menú principal*, serán relacionadas a la denuncia que aquí se encuentre.

3) *Listado de denuncias*, mostrará un listado, en bloques de 15 denuncias, de las quejas o reclamaciones ingresadas en el rango seleccionado en las opciones de *Mes* y *Año*, para pasar una denuncia al área de *denuncia activa*, hacer clic en e número de expediente deseado.

4) *Indicadores de color*, muestra información relacionada con los plazos que se han vencido en el sistema, este color aparecerá en el *listado de denuncias*, a un lado de cada número de expedientes.

5) *Área de Mensajes*, el sistema mandará avisos de mensajes para que el usuario atienda las peticiones que tiene el mismo, ya sea para notificar algún plazo vencido, como para tener un acceso más rápido a las opciones urgentes por realizar del número de expediente relacionado.

6) *Área de Identificación*, muestra el nombre completo del usuario conectado.

3.6.2 Ingreso de denuncias.

Para ingresar una denuncia será seleccionar en el menú principal la siguiente ruta: ***Denuncias –>Ingresar*** (Figura 3.6.3).

Figura 3.6.3 Pantalla que muestra la manera de ingresar una nueva denuncia en el sistema.

Una vez seleccionada esta opción se desplegarán los formularios para que se llenen con la información necesaria para poder dar de alta una nueva denuncia.

Peticionario.

En el formulario de la Figura 3.6.4, se capturarán los datos generales de la persona que interpone la denuncia, también cuenta con un espacio reservado para registrar si cuenta con algún apoderado legal o en caso de no querer recibir notificaciones en su domicilio, puede proporcionar uno distinto en la sección de *Oír y recibir.*

1) Si se diera el caso de que el peticionario es el *Mismo agraviado*, marca esta opción, automáticamente se bloque la pestaña correspondiente al agraviado, para evitar la doble captura de datos.

2) Si fueran más de un peticionario o fuera colectiva la queja o reclamación, marcar la opción en Denunciantes que dice *Sí*, y se activara el campo de *Denunciantes*, donde se escribirán los nombres de los demás peticionarios o el grupo, asociación o comitiva a la que pertenecen.

3) En caso de que el peticionario manifieste que existen más personas autorizadas por él para *oír y recibir* notificaciones, en estos campos se captura esa información.

4) Si el peticionario tiene un *Representante Legal* en esta sección se capturarían los datos correspondientes.

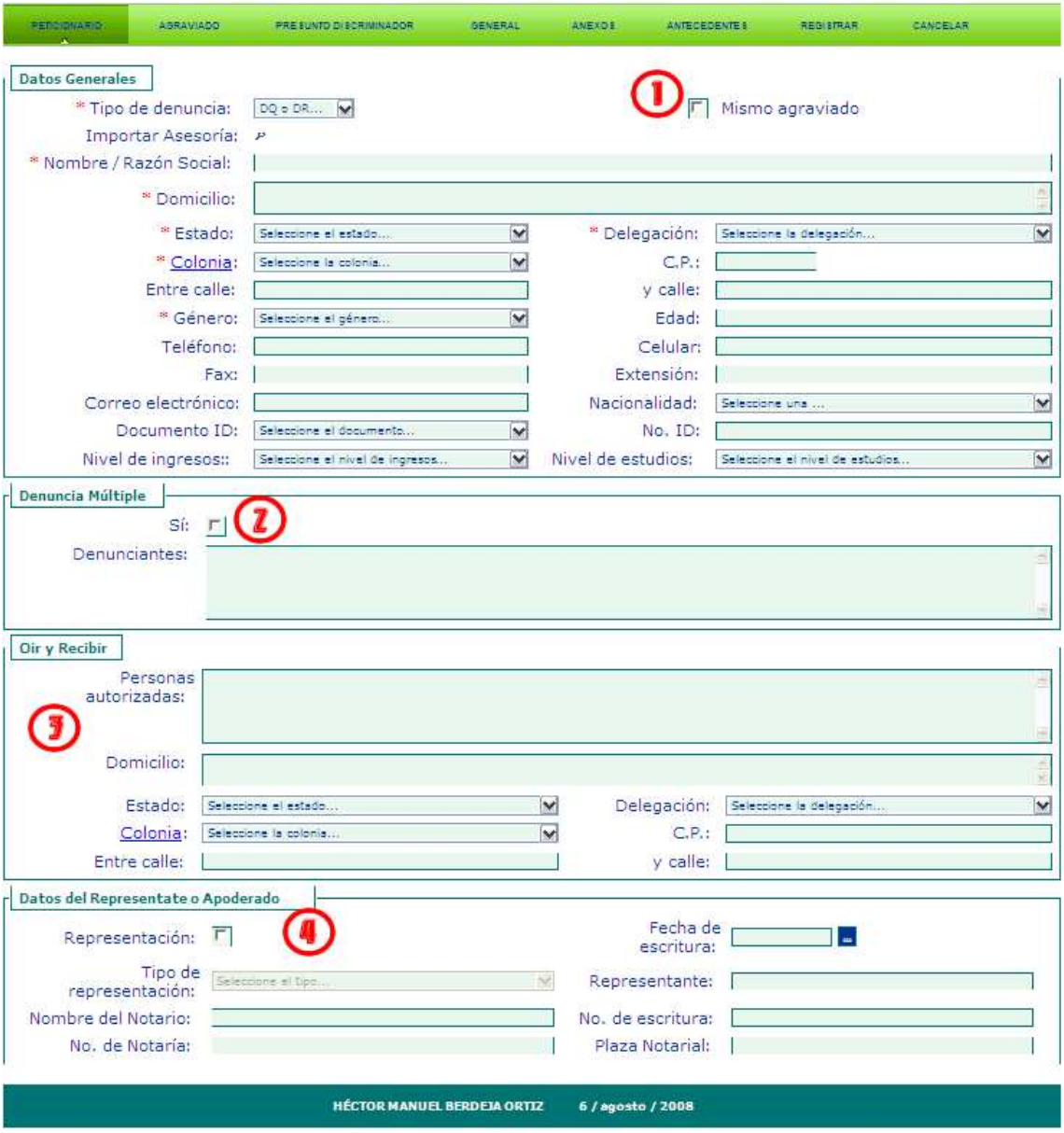


Figura 3.6.4 Formulario de ingreso de datos del Peticionario.

Agraviado.

Los únicos campos relevantes de este formulario (Figura 3.6.5) y por ende son obligatorios son el nombre y el genero del agraviado, en caso de que sea distinto al peticionario.

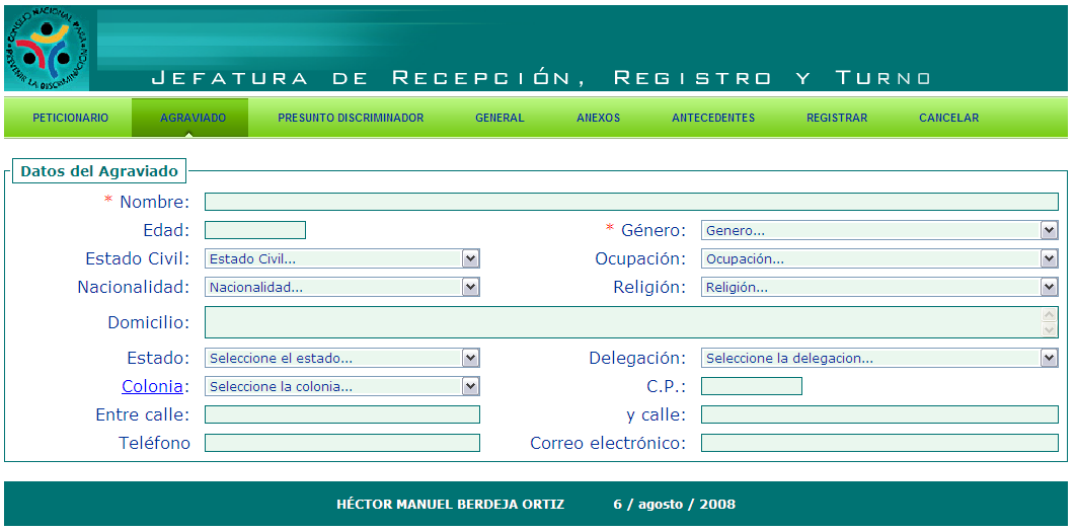


Figura 3.6.5 Formulario de ingreso de datos del Agraviado.

Presunto discriminador.

En el formulario de la Figura 3.6.6, se muestran los campos por capturar para esta persona jurídica dentro del procedimiento del ingreso de la denuncia.

Al seleccionar en el campo *Tipo*, la opción de *Autoridad*, se activarán los campos de *Tipo de Autoridad*, *Autoridad* y *Área Involucrada*. (1)

Figura 3.6.6 Formulario de ingreso de datos del Presunto discriminador.

General.

El *Estado* que hay que capturar en el formulario de la Figura 3.6.7, es el que se refiera al lugar donde se llevaron a cabo los presuntos actos discriminatorios, éste es un paso importante, ya que con este estado se formará el número de expediente final *(1)*.

Figura 3.6.7 Formulario de ingreso de datos generales de la denuncia.

Anexos.

En el caso de que la denuncia venga acompañada de documentos aparte de la inconformidad, éstos serán considerados anexos y se capturarán en la pantalla que muestra la Figura 3.6.8.

Figura 3.6.8 Formulario de ingreso de Anexos a la denuncia.

Antecedentes.

Los antecedentes serán aquellas denuncias previas que se hayan radicado anteriormente del mismo peticionario, esto servirá de referencia para los integradores de expedientes, en caso de existir se agregarán en un formulario como el que detalla la Figura 3.6.9.

Figura 3.6.9 Formulario de ingreso de los antecedentes de la denuncia.

Al hacer clic en se desplegará la ventana de la Figura 3.6.10, en la que se realizará una búsqueda por nombre de peticionario, para verificar si se han iniciado quejas o reclamaciones anteriores y agregarlas a los antecedentes. (1)

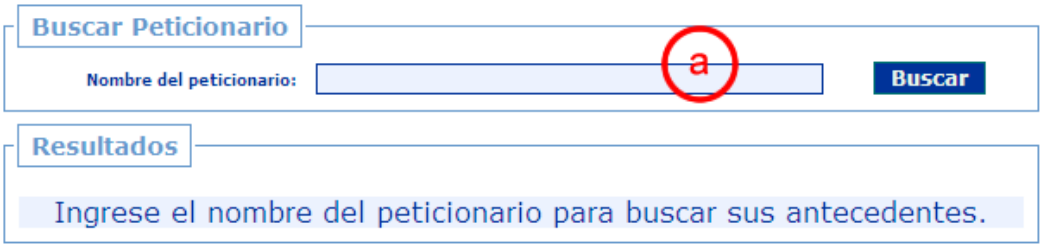


Figura 3.6.10 Pantalla de búsqueda de antecedentes por peticionario.

Ingresa el *nombre del peticionario* que capturó en la pestaña *Peticionario* y realiza la búsqueda *(a)*.

Figura 3.6.11 Pantalla de resultados de la búsqueda de antecedentes por peticionario.

Por ejemplo, buscar al peticionario *Juan*, y dar clic en *Buscar (a)* como se observa en la Figura 3.6.11.

Marcar los registros que correspondan al *Juan* que se acaba de capturar en la pestaña del *Peticionario*, si es que existe, en caso contrario, se entiende que no ha presentado una denuncia anteriormente y los antecedentes quedarán vacíos *(b)*.Figura 3.3.11.

Una vez seleccionados los registros correspondientes damos clic en *Enviar* para que nos plasme estos números de expedientes en la sección de *Antecedentes* del formulario *(c)*. Figura 3.3.11.

En este momento se entiende que los formularios están completos con la información proporcionada por el peticionario, en caso de que se omita un campo obligatorio (marcados con un *), no podrá registrar la actuación, en su caso mostrará una pantalla de error (Figura 3.6.12) con los campos que faltan por capturar. Ya que se ingresen los datos faltantes podrá registrar la denuncia.

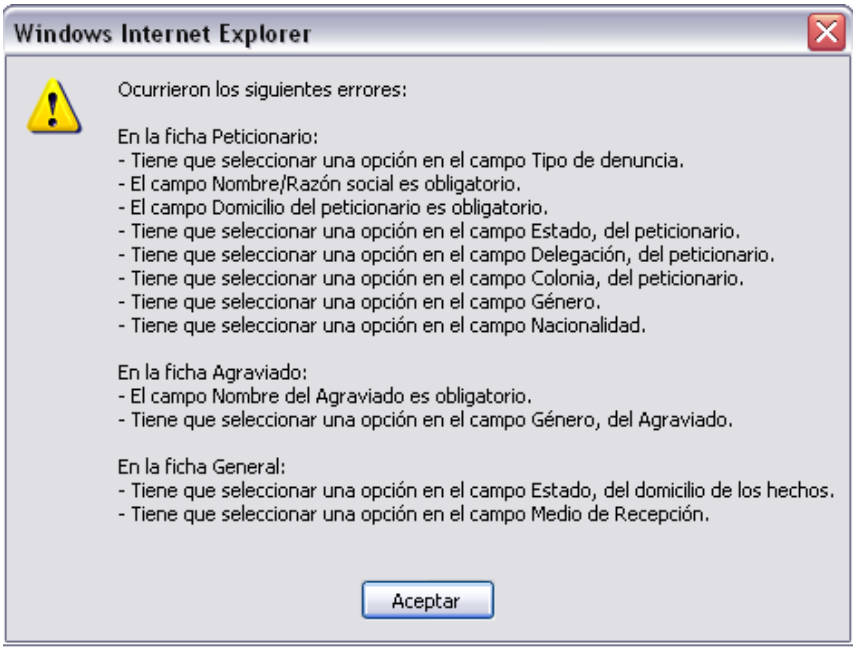


Figura 3.6.12 Pantalla de errores de los formularios.

Registrar.

Figura 3.6.13 Pantalla que muestra una vista previa del Acuerdo de Recepción

Se muestra un previo de lo que contendrá el *Acuerdo de Recepción*, se podrá editar lo necesario para que el documento se imprima de manera correcta como se ve en la Figura 3.6.13, una vez registrada *%%folio%%*, será sustituida por el número de expediente asignado *(1)*.

Registra la denuncia y genera el nuevo número de expediente *(2)*. Figura 3.6.13.

En caso de existir algún error en la captura de los datos, o de no ser necesaria la radicación de este expediente puede cancelar la operación *(3)*. Figura 3.6.13.

3.6.3 Confirmación de registro de denuncias.

Este paso en el sistema permite verificar que la denuncia se haya ingresado correctamente, por lo que una vez que nos muestra el Acuerdo de Recepción, nos mandará a la pantalla de la Figura 3.6.14, desde la cual podremos hacer:

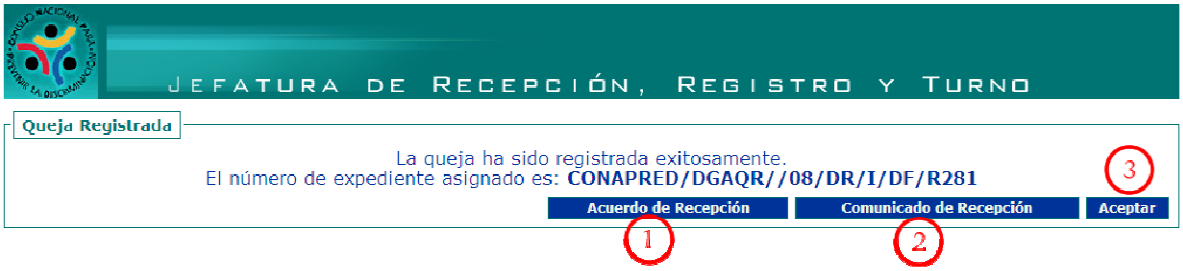


Figura 3.6.14 Pantalla de confirmación de ingreso de denuncias.

Abre el *Acuerdo de Recepción* para su impresión, puede imprimirse posteriormente *(1)*.

Abre el *Comunicado de Recepción* para su impresión, puede imprimirse posteriormente *(2)*.

Confirma el registro y regresa a la *pantalla principal (3)*.

3.6.4 Imprimir carátula de expediente.

Para imprimir este documento, en el menú principal (Figura 3.6.15), se sigue la siguiente ruta: ***Denuncias –> Expediente –> Carátula***.

Figura 3.6.15 Manera de ingresar a imprimir la carátula desde la pantalla principal.

A continuación abrirá una ventana (Figura 3.6.16), donde capturará la palabra correcta (peticionario, peticionaria o agraviado, agraviada).

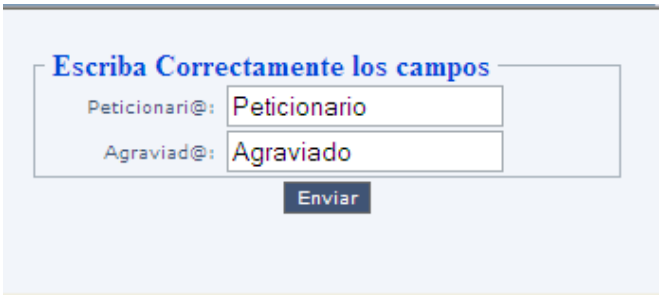


Figura 3.6.16 Ventana de corrección de campos para la carátula.

Al hacer clic en *Enviar* mostrará el documento con los datos previamente capturados y tendremos lista la carátula del expediente.

3.6.5 Imprimir y editar Acuerdo de Recepción.

Como se mencionó anteriormente este documento se puede generar después de confirmar el registro de la queja o reclamación, para obtenerlo se sigue la siguiente ruta: ***Denuncias –> Expediente –> Acuerdo de Recepción –> Acuerdo***, tal como se muestra en la Figura 3.6.17.

Figura 3.6.17 Pantalla que muestra como ingresar a la impresión y edición de los acuerdos.

Para imprimir este documento, aparecerá una pantalla con el acuerdo listo para su impresión (Figura 3.6.18).

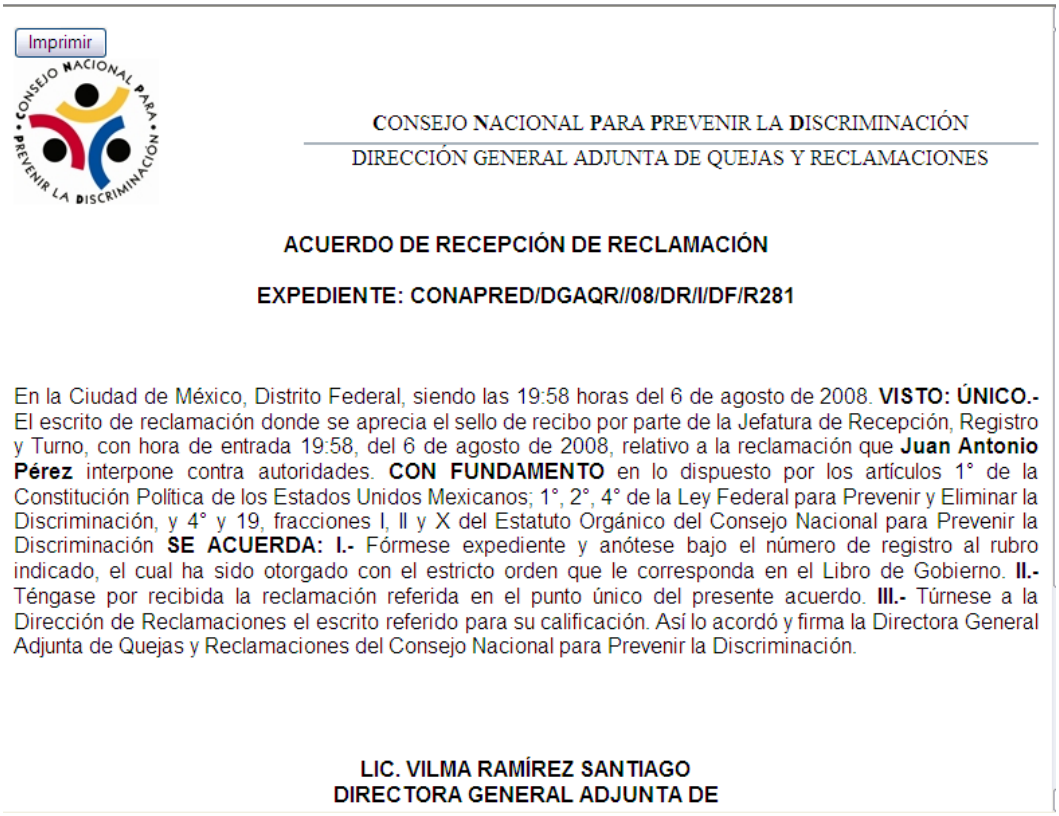


Figura 3.6.18 Documento imprimible del Acuerdo de Recepción.

Edita el contenido del *Acuerdo de Recepción*, y permite modificar los datos del documento como se puede observar en la Figura 3.6.19.

Figura 3.6.19 Pantalla de edición e impresión del Acuerdo de Recepción.

En esta pantalla se pueden modificar los datos de la queja o reclamación, así como el contenido o la redacción del acuerdo, ya que contenga los datos correctos lo se pueden imprimir *(a)*. Después de imprimirlo, hay que *actualizar* el contenido para no perder los cambios que se hicieron *(b)*. Y por último si no se quiere realizar ningún cambio a los datos capturados, podemos regresar al *menú principal (c).*

3.6.6 Imprimir el Comunicado de Recepción.

También es posible imprimir después de la conformación de registro el *Comunicado de Recepción* (Figura 3.6.20), hay que seguir la siguiente ruta y aparecerá el formato imprimible:
Denuncias –> Expediente –> Comunicado de Recepción

Figura 3.6.20 Forma de acceso desde la pantalla principal al Comunicado de Recepción

3.6.7 Turnar una queja o reclamación.

Se podrá realizar el turno de una queja o reclamación, siguiendo la siguiente ruta***: Denuncias –> Turnar***, y aquí debe seleccionar la dirección a la cual se va a turnar (Figura 3.6.21). En caso de que se trate de una queja y se quiera turnar a la Dirección de Reclamaciones, no podrá hacerlo, el sistema simplemente no realizará ninguna acción.

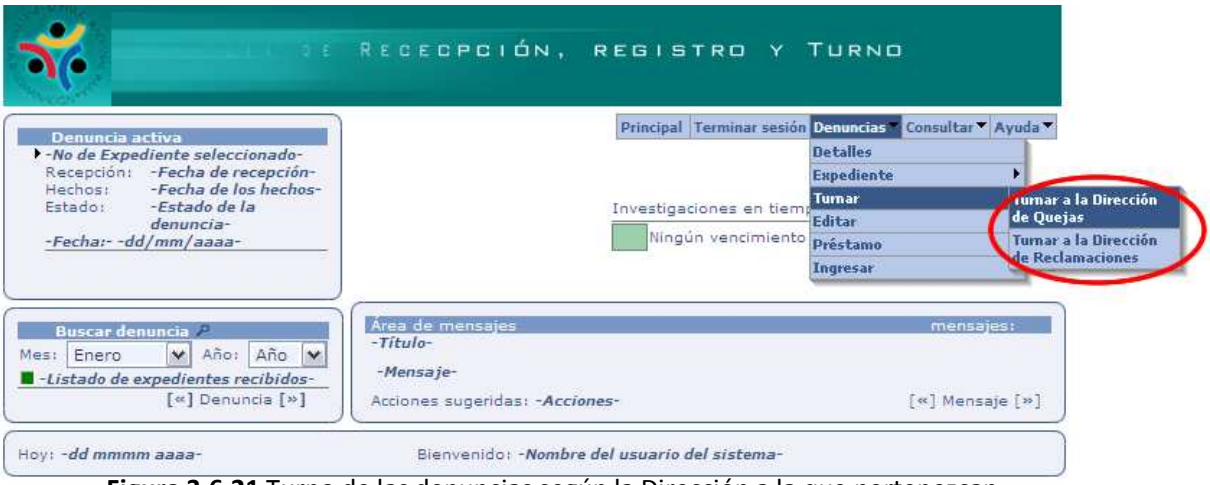


Figura 3.6.21 Turno de las denuncias según la Dirección a la que pertenezcan.

Cuando seleccione la Dirección deberá llenar el formulario que se muestra en la Figura 3.6.22.

Figura 3.6.22 Formulario de turno de denuncias e impresión del Acuerdo de Turno.

Seleccionando un *Responsable* se activará el botón de *Vista de Impresión*, donde se podrá obtener el documento correspondiente al turno del expediente *(1)*, el cual se puede observar en la Figura 3.6.23.

CONSEJO NACIONAL PARA PREVENIR LA DISCRIMINACIÓN
DIRECCIÓN GENERAL ADJUNTA DE QUEJAS Y RECLAMACIONES

DIRECCIÓN DE -QUEJAS / RECLAMACIONES-
JEFE DE RECEPCIÓN REGISTRO Y TURNO –ARCHIVO–

Recibí del Jefe de Recepción, Registro y Turno el expediente No. -*No de expediente*- relativo al presunto agraviado / a la presunta agraviada- -**Nombre del peticionario / agraviado**-, cuyo escrito de fue presentado a este Consejo Nacional para Prevenir la Discriminación por -el peticionario / la peticionaria-- -**Nombre del peticionario(a) / agraviado(a)**-.

El Jefe de Departamento de -*Conciliación / Investigación*- cuyo nombre y firma aparecen al calce, será el responsable de la tramitación de dicho expediente hasta que sea concluido por cualquiera de las causales que dicta este Consejo Nacional para Prevenir la Discriminación. Una vez concluido el expediente, deberá enviarse al archivo para su guarda y custodia.

ATENTAMENTE

México, D. F. a -*dd*- de -*mmm*- de -*aaaa*-.

-*Nombre del responsable*- Hora:_____.
NOMBRE DEL JEFE DE DEPARTAMENTO DE -*CONCILIACIÓN / INVESTIGACIÓN*

_____
FIRMA.

ANEXOS: -*Descripción de los anexos que se presentaron junto con la queja o reclamación*-
No. de documentos. -*no. de documentos que se recibieron*-
No. de fojas. -*no. de fojas de los documentos que se recibieron*-

xxx/xx

Figura 3.6.23 Formato imprimible del Acuerdo de Turno de la denuncia.

Una vez que se imprimió el *Acuerdo de Turno*, se hará clic en *Continuar*, para seguir con el procedimiento del turno *(2)* Figura 3.6.22, y capturar los datos que se piden en el formulario de la Figura 3.6.24.

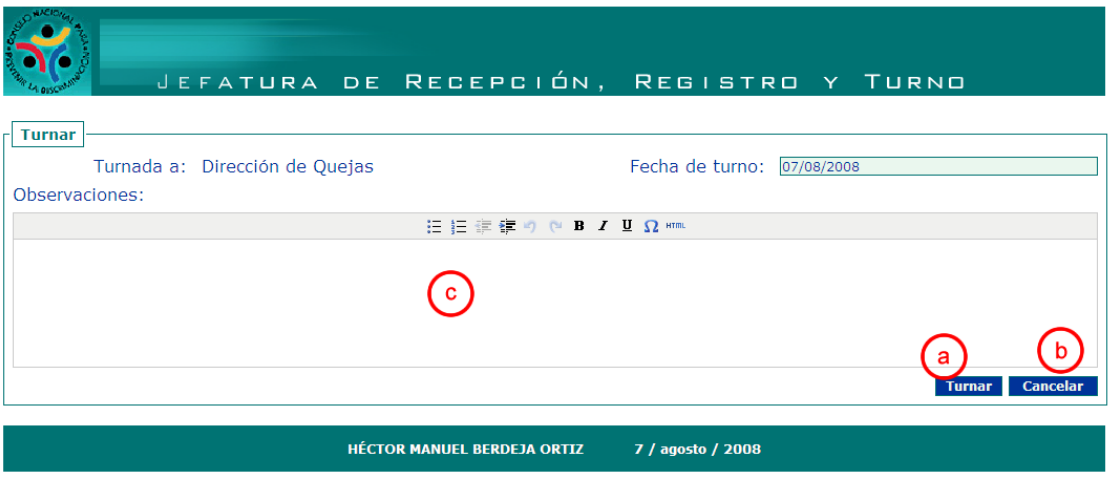


Figura 3.6.24 Formulario de captura de observaciones al turno realizado.

a) Para concluir con el turno del expediente presionar *Turnar*.

b) Si no se desea turnar a esa persona, en ese momento o no es el expediente que se requiere turnar, presione *Cancelar* y lo regresará al *menú principal*.

c) En este espacio podrá hacer las observaciones que considere pertinentes.

En caso de querer cancelar el procedimiento de turno, presione *Cancelar* y el sistema lo regresará al *menú principal (3)* Figura 3.6.22.

3.6.8 Editar denuncias.

Para editar una queja o reclamación, es importante seleccionar primero la denuncia del *listado de denuncias* del lado izquierdo de la pantalla, para posicionarla en el área de *denuncia activa*, una vez seleccionada la queja o reclamación que se quiere editar, hay que acceder a la siguiente ruta: ***Denuncias –> Editar***. (Figura 3.6.25).

Al ingresar, encontrará formularios como los de captura donde modificará el contenido de los datos que desee para después actualizar el registro. Los formularios aparecerán llenos con los datos ingresados en un inicio, los cuales serán tomados de la base de datos.

Estos formularios son exactamente iguales a los de ingreso, lo único que cambia es la presentación del color, para definir cuando se está editando (Figura 3.6.26).

Figura 3.6.25 Ingreso a la edición de denuncias desde la pantalla principal.

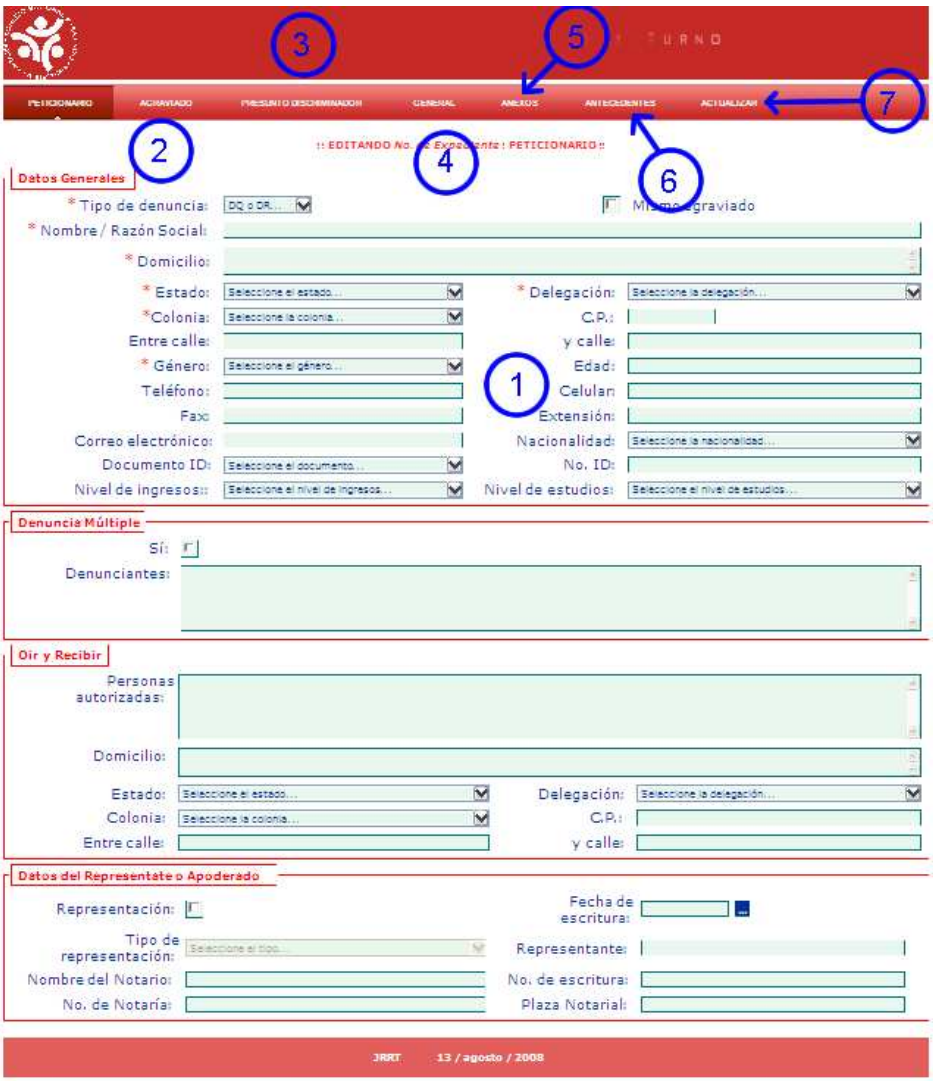


Figura 3.6.26 Formulario de edición de denuncias.

1) El primer formulario, es el que contiene los datos del *peticionario*.
2) El formulario del *Agraviado*.
3) El formulario del *Presunto Discriminador*.
4) El formulario *General*, donde se capturan los datos del domicilio del presunto acto de discriminación.
5) El formulario de los *Anexos*.
6) El formulario de los A*ntecedentes*.
7) Al concluir de editar los datos necesarios, sólo hay que presionar *Actualizar*, para que quede constancia de los cambios realizados.

3.6.9 Detalles de la denuncia.

Se pueden consultar los detalles de las quejas o reclamaciones, los cuales, serán los datos ingresados por la Jefatura de Recepción, Registro y Turno, para verlos hay que acceder a la siguiente ruta: ***Denuncias –> Detalles***, como aparece en la Figura 3.6.27.

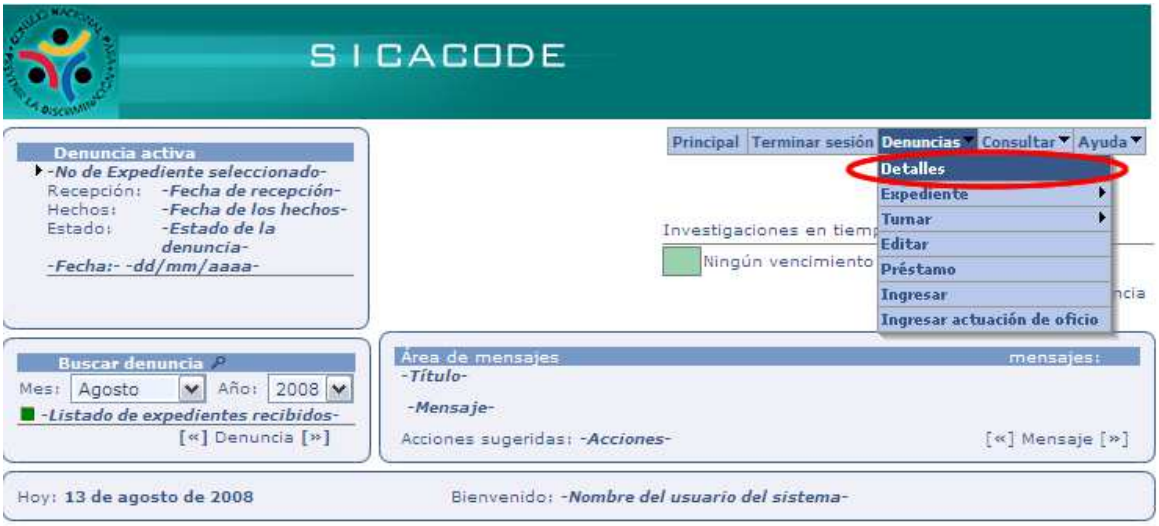


Figura 3.6.27 Acceder a los detalles de las denuncias desde la pantalla principal.

A continuación aparecerá una pantalla con pestañas indicando los datos capturados para cada uno de los que intervienen en el procedimiento de la queja o reclamación. El formulario de detalles es igual al de captura y al de edición, lo que los distinguirá serán los colores (Figura 3.6.28), y éste en especial no podrá hacer modificaciones de ningún tipo, es únicamente de consulta. Dentro de los elementos que componen esta pantalla están:

1) Pantalla de los datos del *Peticionario*.

2) Pantalla de los datos del *Agraviado*.

3) Pantalla de los datos del *Presunto Discriminador*.

4) Pantalla de los datos *generales del lugar de los hechos*.

5) Pantalla de los *Anexos* presentados.

6) Pantalla de *Antecedentes* del peticionario en el Consejo.

7) Una vez terminada la consulta de estos datos puede *Regresar* a la *pantalla principal*.

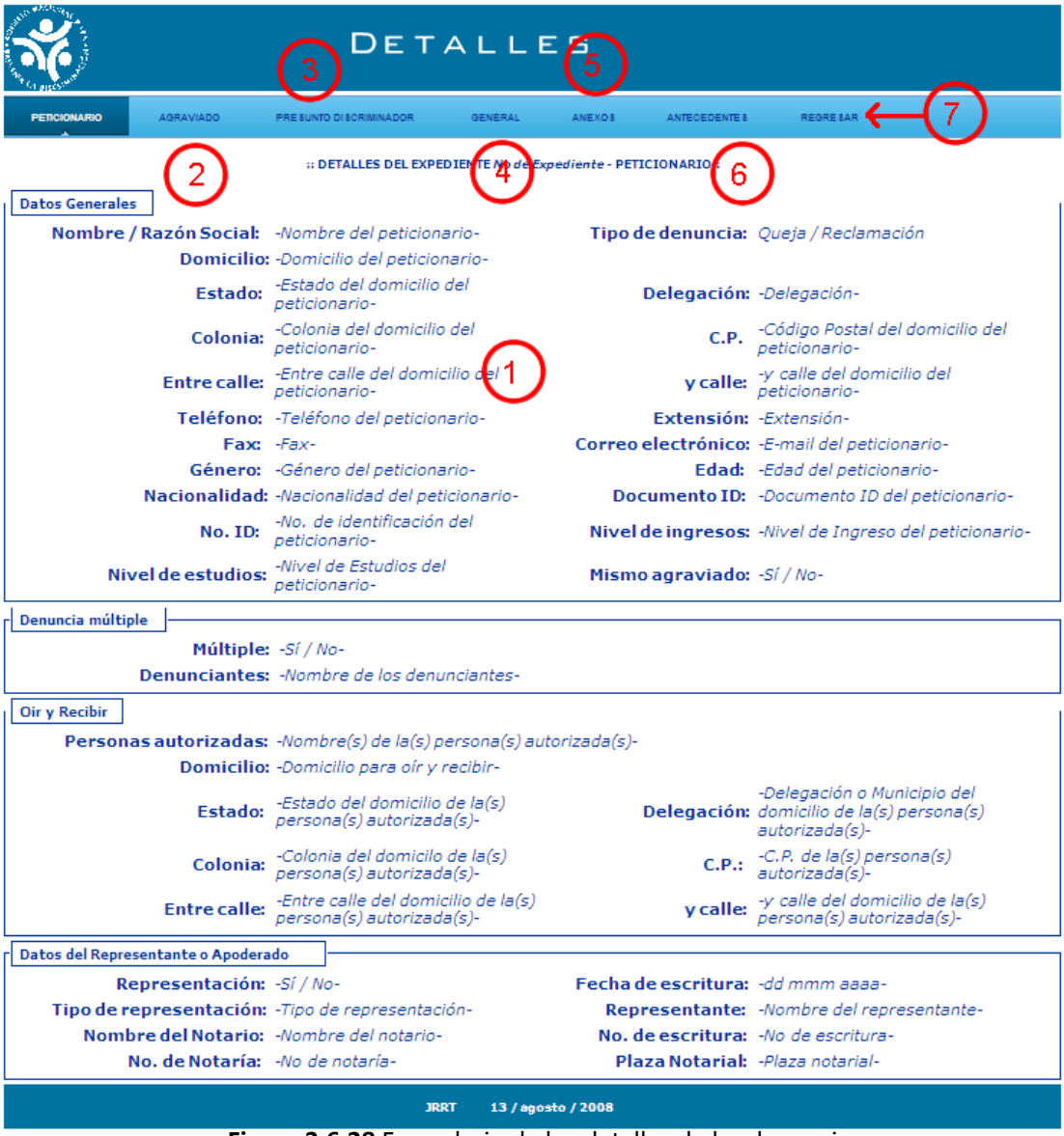


Figura 3.6.28 Formulario de los detalles de las denuncias.

3.6.10 Buscar denuncias.

El sistema también permite la búsqueda de denuncias, por medio de un formulario donde se muestran los filtros para encontrar los datos de la queja o reclamación que se está buscando (Figura 3.6.29), sólo bastará con llenar los campos que se consideren necesarios. En caso de no haber resultados mostrar el formulario nos mostrará este mensaje.

Figura 3.6.29 Formulario de búsqueda de denuncias

3.6.11 Préstamo de expedientes.

Una de las acciones que lleva a cabo el personal de la Jefatura de Recepción, Registro y Turno es la del préstamo de expedientes, con la finalidad de llevar un control de los de a quién y en que fecha se prestaron, se elaboró este formulario para capturar y guardar los datos pertinentes (Figura 3.6.30).

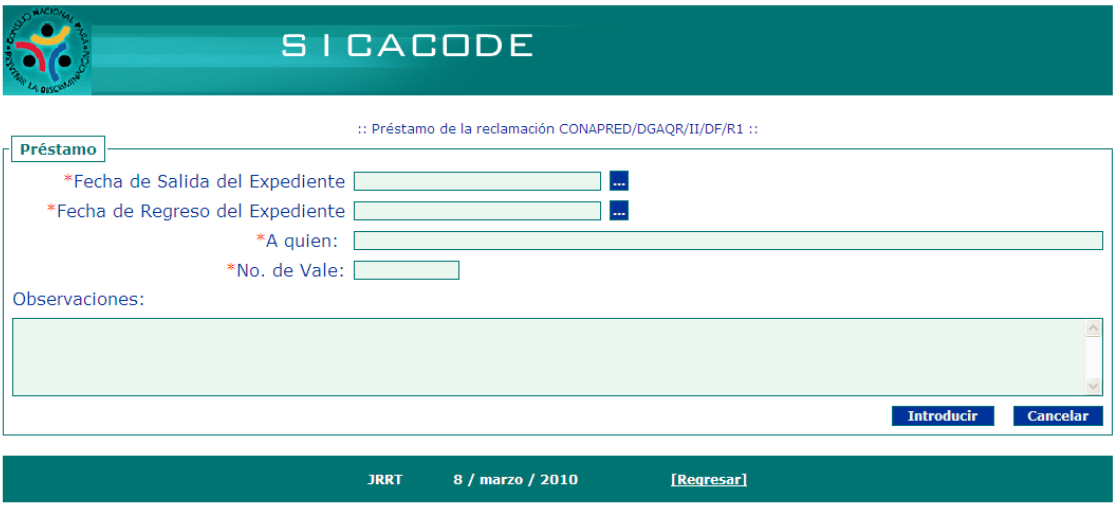


Figura 3.6.30 Formulario de préstamo de expedientes.

3.7 Configuración del servidor WWW, Apache.

El *SICACODE* pretende ser un sistema que se maneje en la intranet de la institución, por lo que la configuración del servidor WWW (Apache), quedará como se explica a continuación, esto con el fin de optimizar el uso del equipo en el que se pretende montar así como para evitar conflictos con otros sistemas que posiblemente se monten posteriormente en el mismo. La versión a utilizar será la 2.2.9, por ser la más estable hasta el momento. Las modificaciones al archivo *httpd.conf* serán las siguientes:

- **ServerRoot** ***/usr/local/apache/apache2***, esta será la ubicación del servidor WWW, dentro del sistema.

- **Listen** ***8080***, el puerto por el que escuchará peticiones el servidor.

- **ServerAdmin** ***yaen.sanders@consejo.intranet.org.mx***, será el administrador del servidor.

- **ServerName** ***conap1.consejo.intranet.org.mx:80***, es el nombre del servidor, que se encuentra dentro del dominio *consejo.intranet.org.mx*.

- **DocumentRoot "*/usr/local/apache/apache2/htdocs/sicacode*"**, es el directorio en donde se alojarán todos los scripts del sistema.

- **DirectoryIndex** ***index.php index.html***, será el archivo que tome por omisión el Apache, esto debido a que el sistema será montado en *PHP*.

- El usuario *consejo* y el grupo *consejo* van a ser lo que puedan levantar el servidor www.

- La línea para habilitar el módulo de PHP que será utilizado para desarrollar el sistema, quedará de la siguiente manera: *Include /etc/httpd/mod_php.conf.*

El sistema esta pensado para tener como máximo 25 usuarios, tomando en cuenta que no se ocuparán demasiadas peticiones simultáneas, el archivo de rendimiento del servidor Apache "*httpd-mpm.conf*" quedará de la siguiente manera:

```
StartServers          5
MinSpareServers       5
MaxSpareServers      15
MaxClients           30
MaxRequestsPerChild   0
```

Como módulos adicionales al sistema se implementará *PHP* y *MySQL*, de los cuales se pretende utilizar las versiones más actuales y estables.

En cuestiones de seguridad los módulos de información y estado (*mod_status* y *mod_info*) del servidor Apache quedarán inhabilitados al momento de la puesta en marcha del sistema.

También, en la seguridad de acceso al sistema, no se realizará con Apache, ésta será manejada por *PHP*, creando sesiones de usuario para subsanar este punto.

3.8 Detalles sobre la programación del proyecto.

El objetivo de este sistema es optimizar el procedimiento de captura, registro y turno de expedientes, por lo que es conveniente implementarlo con la metodología *Cliente-Servidor*, para liberar el espacio localmente en los equipos de cómputo.

La Dirección General Adjunta de Quejas y Reclamaciones cuenta con dos direcciones, por lo tanto existen dos procedimientos que, aunque parecidos, no son iguales, es por esto que se opta por desarrollar la programación con PHP, por las siguientes cuestiones:

- Facilidad de manejo e implementación.
- Por que ofrece una programación netamente dinámica, facilitando el desarrollo de los procedimientos.
- Por la portabilidad que ofrece, al ser un lenguaje multiplataforma.
- Por la rapidez de ejecución utilizando muy poca memoria RAM.
- Capacidad de conexión con la mayoría de los manejadores de bases de datos que se utilizados actualmente.
- Trabaja perfectamente con el Servidor Web, Apache.
- Permite crear la presentación con HTML, permitiendo la elaboración de interfases llamativas para el usuario.

3.9 Descripción modular del proyecto.

La descripción modular del sistema consta de varios elementos a considerar: el módulo de la Jefatura de recepción, Registro y Tuno (Figura 3.9.1); el del administrador (Figura 3.9.2); y el de los scripts que se incluyen en todos los archivos del sistema y que son de gran relevancia para el funcionamiento del sistema (Figura 3.9.3). Los archivos de menor relevancia se comentarán en donde aparezcan.

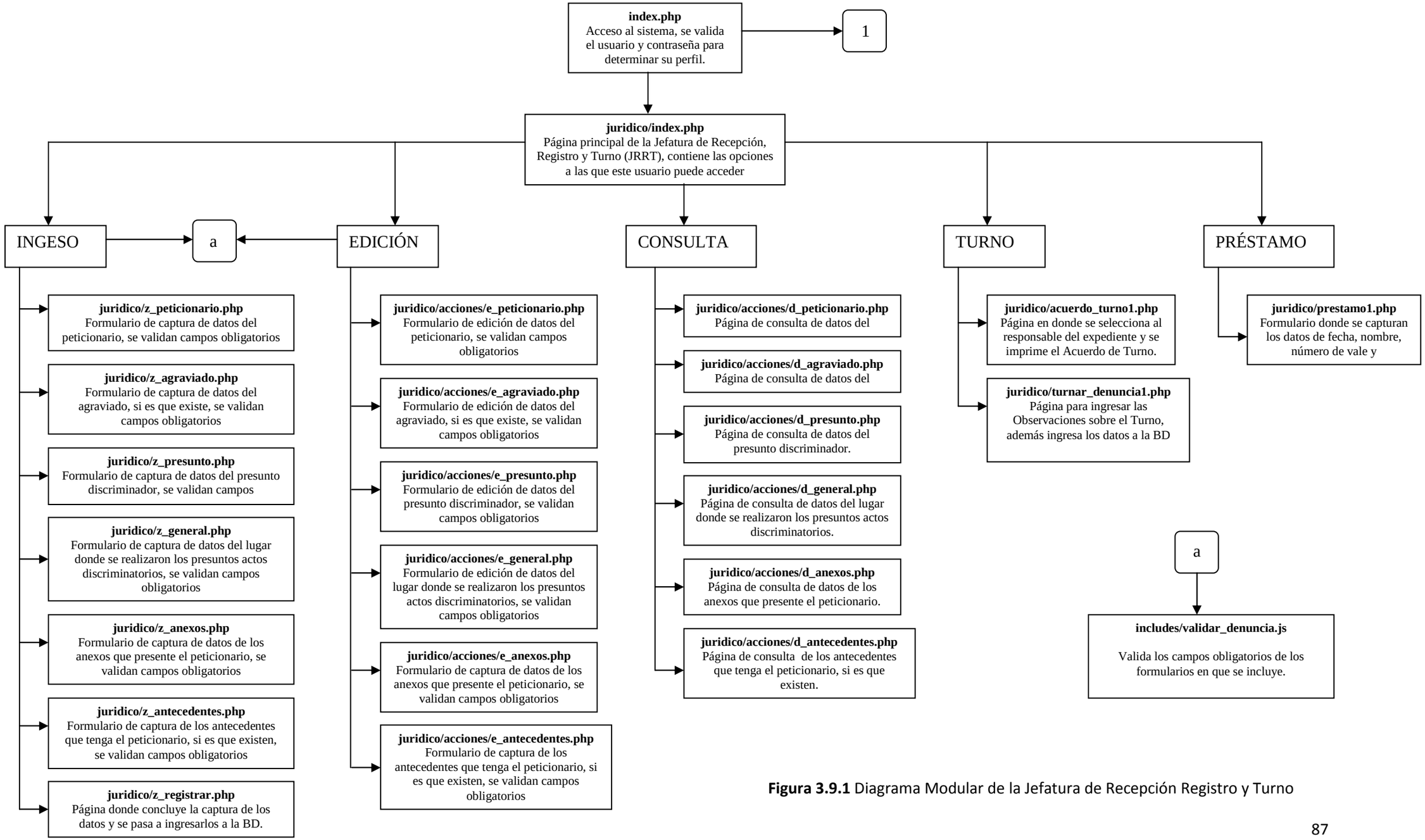


Figura 3.9.1 Diagrama Modular de la Jefatura de Recepción Registro y Turno

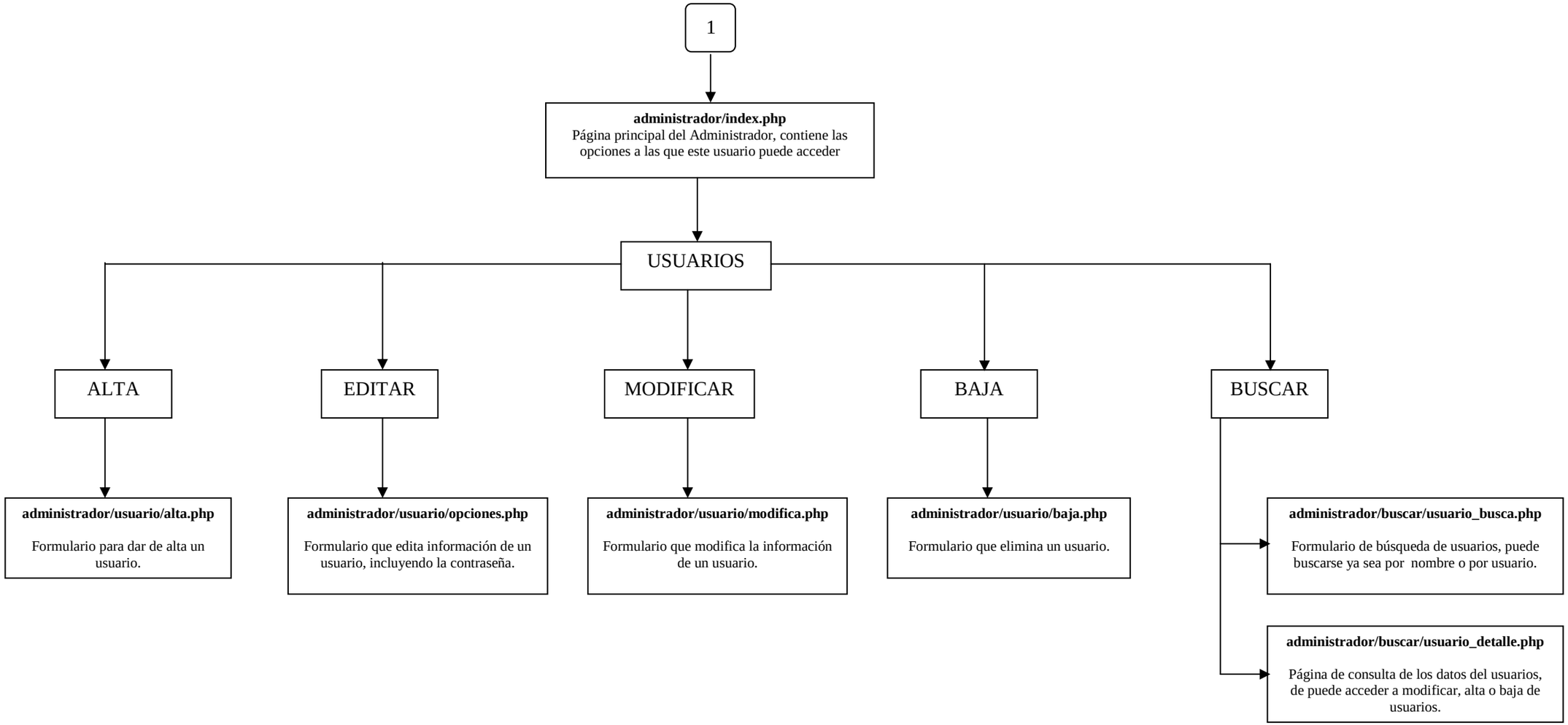


Figura 3.9.2 Diagrama Modular del Administrador del sistema.

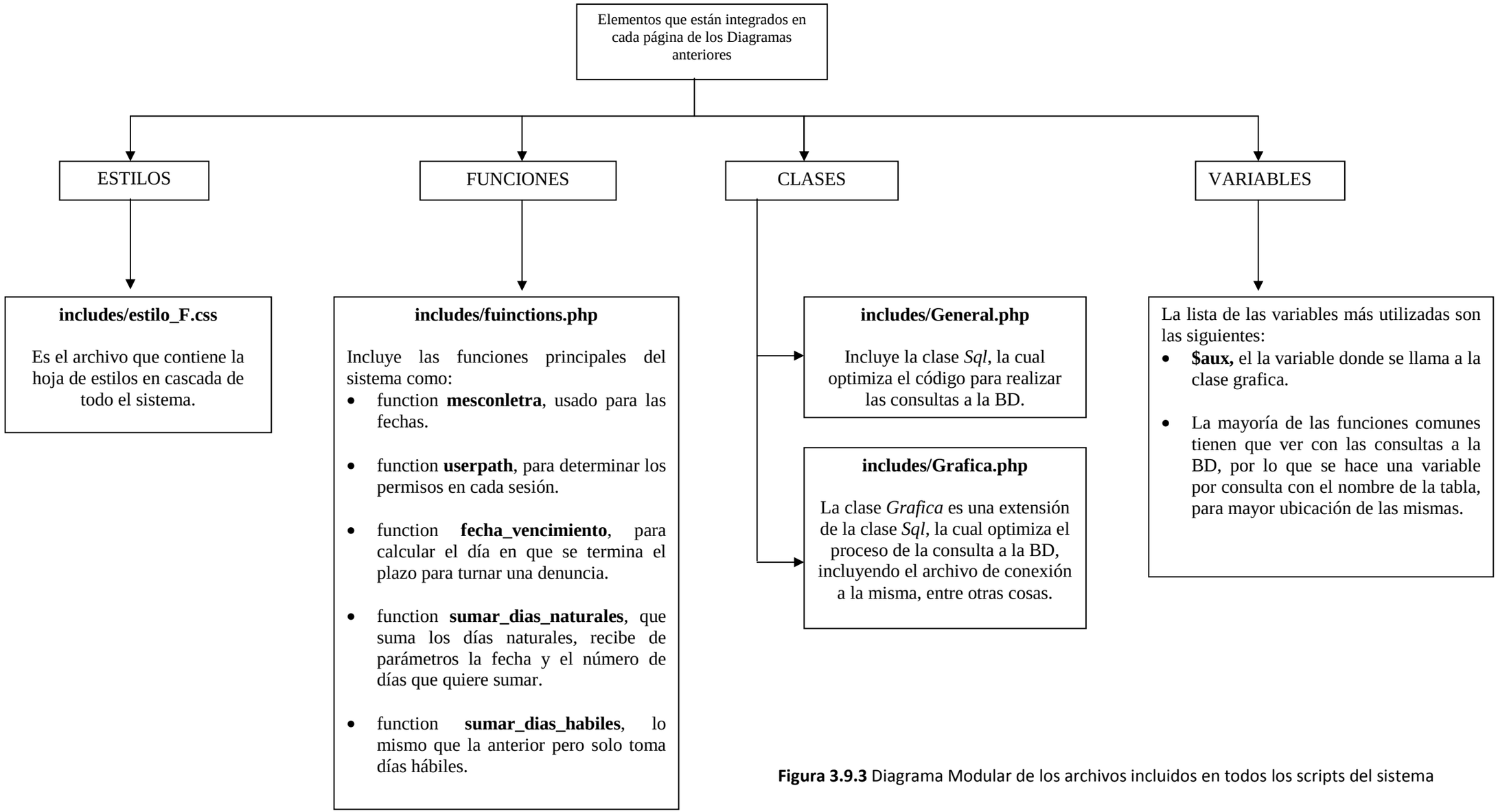
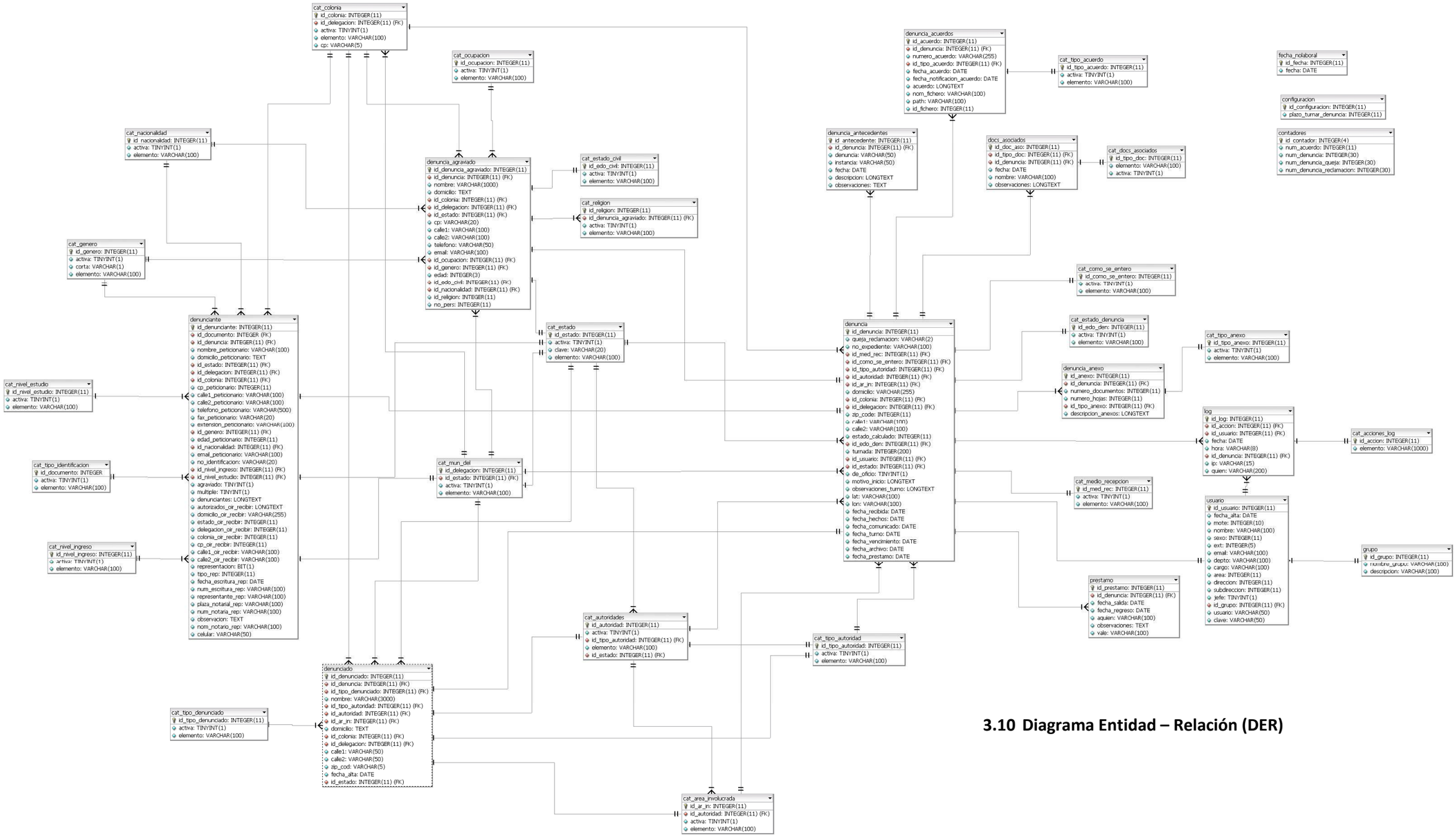


Figura 3.9.3 Diagrama Modular de los archivos incluidos en todos los scripts del sistema



3.10 Diagrama Entidad – Relación (DER)

3.11 Descripción del método de conexión de *PHP* con *MySQL* en el sistema.

El método de conexión se realiza mediante el archivo ubicado en *public_html/Connections/cnn_conapred.php*.

Este archivo contiene el nombre del *host*, la *Base de Datos* y el *password* de la misma, básicamente la conexión con la base de datos la hace mediante la clase *adodb* y todos sus archivos están concentrados en una carpeta con este mismo nombre. Esta clase facilitará varios procedimientos como por ejemplo:

- Conexión a la *BD*.
- Publicación de los campos de una consulta a la *BD*, de una manera más fácil y práctica.
- Mejor manejo de *Recordsets*.
- Soporta varios tipos de bases de datos, entre ellas *MySQL*.
- En general, un fácil manejo de todos los aspectos relacionados que tienen PHP y *MySQL* (*Connect, PConnect, NConnect, Execute, CacheExecute, SelectLimit, CacheSelectLimit, MoveNext, Close, qstr, Affected_Rows, Insert_ID*).

Este *script* de conexión se incluirá en todas las pantallas del sistema para que se pueda tener acceso a la Base de Datos, y así, consultarla o ingresar, actualizar o borrar datos de la misma.

Es así como esta conformado el proyecto que se presentó para acreditar el Diplomado en cuestión, y como se puede observar implementa todos los temas vistos. En términos generales es un sistema integral al cual se le podrían agregar funcionalidades, aunque cabe mencionar, que éste es sólo una parte de un sistema más completo que se desarrolló para la institución mencionada y el cual aún se encuentra en uso.

En conclusión se cumplió con lo establecido en los objetivos planteados desde un principio y sin embargo es material sujeto de mejora continua, aunque para los fines que se realizó cumple con las expectativas esperadas.

4. Conclusiones generales.

En mi opinión, los contenidos vistos en los cursos y el diplomado fueron bastante atractivos y bien explicados por los instructores, además de que se ampliaron los conocimientos y mejoraron las técnicas de programación en el diseño de sistemas.

Al elegir esta forma de titulación, lo que también pretendía era extender mi panorama sobre la computación y sus alcances actuales, que son muy amplios, así como la actualización en los temas referentes a ella; con el diplomado y los cursos se cumplió la expectativa e incluso el interés por la capacitación continua creció, personalmente hablando.

El darse cuenta de que existen herramientas eficaces y eficientes en la operación y tratamiento de la información que además de todo no tienen un precio de licenciamiento, es una ventaja al momento de enfrentarse con problemas de la vida laboral real, ya que, sin duda, ofrecen una solución bastante viable y con grandes resultados; aunque como todo existen limitaciones, pero hay que saber implementar lo que esta hecho y adaptarlo a nuestras necesidades, ahí es donde se pueden dejar ver las técnicas adquiridas en los cursos antes descritos.

El cursar estas materias también impulsa a seguir con la actualización en cuestiones computacionales y de tecnología que pueden ser explotadas y mejoradas en estos tiempos lo cual me parece un muy buen incentivo para continuar.

Glosario

Sistema Operativo.

Es un software que actúa de interfaz entre los dispositivos de hardware y software para utilizar un computador. Es responsable de gestionar, coordinar las actividades y llevar a cabo el intercambio de los recursos del ordenador y actúa como intermediario para las aplicaciones que se ejecutan.

Linux.

El núcleo Linux es un sistema operativo libre tipo Unix. Es uno de los principales ejemplos de software libre y código abierto. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo. El desarrollo del día a día tiene lugar en la *Linux Kernel Mailing List.*

Software libre.

Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente. Según la *Free Software Foundation*, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.

Metacaracteres.

Una expresión regular es una cadena que tiene ciertos caracteres con un significado especial, y con la que podremos referirnos a múltiples ficheros. Esos caracteres especiales se denominan caracteres *comodín* o *metacaracteres*, y son interpretados por la *shell* antes de ser ejecutado el comando. Son tres: el asterisco (*), la interrogación (?) y los corchetes ([ ]).

FTP.

File Transfer Protocol (Protocolo de Transferencia de Archivos), es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (*Transmission Control Protocol*), basado en la arquitectura *cliente-servidor*. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

SSH.

Secure Shell, en español, intérprete de órdenes segura. Es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también

puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X (en sistemas Unix y Windows) corriendo.

Además de la conexión a otros dispositivos, permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a los dispositivos y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante *SSH*.

Unix

Es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios *Bell* de *AT&T*, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

GNU

Es un acrónimo recursivo que significa *GNU No es Unix* (GNU is Not Unix). Puesto que en inglés "gnu" (en español "ñu") se pronuncia igual que "new", Richard Stallman recomienda pronunciarlo "guh-noo". En español, se recomienda pronunciarlo ñu como el antílope africano o fonéticamente; por ello, el término mayoritariamente se deletrea (G-N-U) para su mejor comprensión. En sus charlas Richard Stallman finalmente dice siempre «Se puede pronunciar de cualquier forma, la única pronunciación errónea es decirle 'linux'».

Kernel

El núcleo o kernel (de la raíz germánica *Kern*) es un software que actúa de sistema operativo. Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

Script

Archivo de órdenes o *archivo de procesamiento por lotes*, es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Casi siempre son interpretados, pero no todo programa interpretado es considerado un *script*. Su uso habitual es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso es frecuente que los *shells* sean a la vez intérpretes de este tipo de programas.

FAT

Tabla de Asignación de Archivos, en inglés, *File Allocation Table,* es un sistema de archivos desarrollado para MS-DOS, así como el sistema de archivos principal de las ediciones no empresariales de Microsoft Windows hasta Windows Me.

Se utiliza como mecanismo de intercambio de datos entre sistemas operativos distintos que coexisten en el mismo computador, lo que se conoce como entorno *multiarranque*. También se utiliza en tarjetas de memoria y dispositivos similares.
Las implementaciones más extendidas de FAT tienen algunas desventajas. Cuando se borran y se escriben nuevos archivos tienden a dejar fragmentos dispersos de éstos por todo el soporte. Con el tiempo, esto hace que el proceso de lectura o escritura sea cada vez más lento. La denominada desfragmentación es la solución a esto, pero es un proceso largo que debe repetirse regularmente para mantener el sistema de archivos en perfectas condiciones. FAT tampoco fue diseñado para ser redundante ante fallos. Inicialmente solamente soportaba nombres cortos de archivo: ocho caracteres para el nombre más tres para la extensión. También carece de permisos de seguridad: cualquier usuario puede acceder a cualquier archivo.

I-nodos

Nodo-i o *nodo índice*, es una estructura de datos propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo *UNIX* como es el caso de *Linux*. Un inodo contiene las características (permisos, fechas, ubicación, pero no el nombre) de un archivo regular, directorio, o cualquier otro objeto que pueda contener el sistema de ficheros.

El término "inodo" refiere generalmente a inodos en discos (dispositivos en modo bloque) que almacenan archivos regulares, directorios, y enlaces simbólicos. El concepto es particularmente importante para la recuperación de los sistemas de archivos dañados.

Cada inodo queda identificado por un número entero, único dentro del sistema de ficheros, y los directorios recogen una lista de parejas formadas por un número de inodo y nombre identificativo que permite acceder al archivo en cuestión: cada archivo tiene un único inodo, pero puede tener más de un nombre en distintos o incluso en el mismo directorio para facilitar

Shell

Se emplea para referirse a programas que proveen una interfaz de usuario para acceder a los servicios del sistema operativo. Estos pueden ser gráficos o de texto simple, dependiendo del tipo de interfaz que empleen. Están diseñados para facilitar la forma en que se invocan o ejecutan los distintos programas disponibles en la computadora.

IDE

Entorno de desarrollo informático (en inglés I*ntegrated Development Environment*) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

El puerto IDE (*Integrated device Electronics*) o ATA (*Advanced Technology Attachment*) controla los dispositivos de almacenamiento masivo de datos, como los discos duros y ATAPI (*Advanced Technology Attachment Packet Interface*) y además añade dispositivos como las unidades CD-ROM. En este sistema el controlador del dispositivo se encuentra integrado en la electrónica del dispositivo

SATA

Acrónimo de *Serial Advanced Technology Attachment* es una interfaz de transferencia de datos entre la placa base y algunos dispositivos de almacenamiento, como puede ser el disco duro, lectores y regrabadores de CD/DVD/BR, Unidades de Estado Sólido u otros dispositivos de altas prestaciones que están siendo todavía desarrollados

IP

Una dirección IP es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP *(Internet Protocol*), que corresponde al nivel de red del protocolo *TCP/IP*. Dicho número no se ha de confundir con la dirección *MAC* que es un número hexadecimal fijo que es asignado a la tarjeta o dispositivo de red por el fabricante, mientras que la dirección IP se puede cambiar. Esta dirección puede cambiar 2 ó 3 veces al día; y a esta forma de asignación de direcciones se denomina dirección IP dinámica.

MBR

Master Boot Record, es el primer sector ("sector cero") de un dispositivo de almacenamiento de datos, como un disco duro. A veces, se emplea para el arranque del sistema operativo con *bootstrap*, otras veces es usado para almacenar una tabla de particiones y, en ocasiones, se usa sólo para identificar un dispositivo de disco individual, aunque en algunas máquinas esto último no se usa y es ignorado.

Gateway o puerta de enlace

Dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación, su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino.

W3C.

El *Consorcio World Wide Web* es un consorcio internacional donde las Organizaciones miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web. La misión del W3C es: *Guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.*

SGML

Standard Generalized Markup Language o Estándar de Lenguaje de Marcado Generalizado. Consiste en un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) normalizó este lenguaje en 1986. El lenguaje SGML sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial. El lenguaje HTML está definido en términos del SGML

HTTP

Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la *World Wide Web*. Desarrollado por el *World Wide Web Consortium* y la *Internet Engineering Task Force*. Es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado, para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente, esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

DNS

Domain Name System / Service (sistema de nombre de dominio) es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada, asocia información variada con nombres de dominios asignado a cada uno de los participantes y su función más importante, es traducir nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.

Demonio

Un demonio, *daemon* o *dæmon* (de sus siglas en inglés *Disk And Execution MONitor*), es un tipo especial de proceso informático que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario (es un proceso no interactivo). Este tipo de programas se ejecutan de forma continua, vale decir, que aunque se intente cerrar o matar el proceso, este continuará en ejecución o se reiniciará automáticamente. Todo esto sin intervención de terceros y sin dependencia de consola alguna.

Firewall

Es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar, descifrar, el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios. Pueden ser implementados en hardware o software, o una combinación de ambos.

Framework

En el desarrollo de software, un *framework* es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Comandos en Linux.

- **mv**: mover archivos
- **cp**: copiar archivos
- **ssh**: Loguearse vía remota a otro máquina
- **cat**: mostrar el contenido de un archivo
- **more**: listar el contenido de un archivo como si estuvieras en un editor de texto.
- **less**: lista las diez últimas líneas de un archivo.
- **chmod**: Cambiar los permisos de archivos
- **chown**: Cambiar el dueño de un archivo
- **useradd**: Agregar un usuario

Niveles de inicio de Linux:

- 0, halt o suspendido.
- 1, modo de usuario simple.
- 2, unused, pero es configurado igual que el 3.
- 3, modo multiusuario,
- 4, X11 o modo gráfico con KDM/GDM/XDM.
- 5, unused, pero es configurado igual que el 3.
- 6, reboot o reinicio.

Los comandos básicos para administra usuarios:

- **Adduser**, agrega usuarios en modo comando.
- **Useradd**, funciona igual que el anterior.
- **Userdel**, elimina un usuario.
- **Usermod**, modifica las propiedades del usuario.
- **Groupadd**, agrega un grupo.
- **Groupdel**, elimina grupos.
- **Groupmod**, modifica un grupo.
- **Groups**, lista los usuarios a los que pertenece un usuario.
- **Passwd**, modifica las propiedades de usuarios y grupos.
- **Kuser**, administra usuarios y grupos en modo gráfico.

Secuencias de una lista en HTML.

- **type="1"**. Listará con números consecutivos.
- **type="A".** Listará con letras mayúsculas consecutivas
- **type="a"**. Definirá una lista con letras minúsculas consecutivas.
- **type="I".** Hará una lista con números romanos.
- **type="i".** Listará con números romanos en minúsculas.

Atributos de la etiqueta <*frameset*>.

- **cols**. con este atributo se indica cuantas columnas se tendrán, por ejemplo *cols="x,y,z*" recibe valores en pixeles o porcentajes.

- **rows**. Se indica cuantas filas se utilizarán, se le asignan los valores de la siguiente manera *rows="x,z,y*" y recibe valores en pixeles o porcentajes.

- **frameborder.** Borde de los *frames*, recibe valores de 0 y 1, 0 no lo muestra y 1 si lo muestra.

- **border**. Indica el ancho del borde del *frame*, recibe valores en pixeles.

- **bordercolor**. Define el color del borde y recibe valores hexadecimales o su equivalente al nombre del color en ingles.

Atributos de la etiqueta <*frame*>.

- **name**. Indica el nombre del frame.
- **src**. Hace referencia a la ruta hacia el documento HTML que se hará referencia.
- **scrolling**. Recibe yes, auto o no, y ayuda a determinar si podremos hacer uso de la barra de desplazamiento pero solo en el área del marco indicado. Por *default* tiene *yes*.

Atributos de la etiqueta *<input>*.

- **type**. es el tipo de *input* que se va a usar, dependiendo del elemento, los tipos existentes son:

    - **radio**. Selector de una y solo una opción.
    - **checkbox**. Casilla de verificación.
    - **hidden**. Es un elemento oculto siendo *value* el único atributo extra que comparta con los demás elementos.
    - **file**. Define un archivo que puede ser enviado junto con los demás elementos del formulario.
    - **submit**. Es un botón que envía nuestro formulario al archivo declarado en el atributo *action* de la etiqueta *form*.
    - **reset**. Borra todo lo seleccionado o escrito en todos los elementos del formulario.
    - **button**. es un botón estándar que por si solo no tienen ninguna utilidad pero que puede dársele con javascript en el atributo *onclick*.
    - **passwd.** Es muy similar al elemento *text* pero se utiliza para los campos de contraseña ya que substituye los caracteres escritos por puntos.

- **value**. Es el valor que tendrá nuestra etiqueta, puede ser cualquiera que determinemos

- **checked**. Recibe valores como *true* y *false*, y solo aplica a elementos de tipo *radio* y *checkbox*.

- **name**. Es el nombre con el cual distinguiremos el elemento del formulario.

- **disable**. Recibe valores de *true* y *false*, cuando esta en *true*, no podremos tomar ninguna acción sobre dicho elemento.

Parámetros de configuración de Apache.

- **StartServers**. Define el número de servidores que arrancarán desde un principio.

- **MinSpareServers**. Es el número mínimo de servidores disponibles.

- **MaxSpareServers**. Número máximo de servidores disponibles.

- **MaxClients**. Define el número máximo de clientes que se pueden atender simultáneamente.

- **MaxRequestPerChild**. Establece el número máximo de peticiones simultáneas por cada cliente atendido.

Funciones útiles de *PHP*.

- **time(*hora*).** Devuelve la hora en formato *UNIX*.
- **date(*fecha*).** Devuelve la fecha con el formato de la hora que le pasemos.
- **array(*campo=>valor*).** Nos permite manejar arreglos, los cuales pueden llenarse de las siguientes maneras:

  ```
  o  $array[0]=1; $array[1]=2; $array[2]=3 …
  o  $array=array(0=>1, 1=>2,…);
  o  $array=array(1,2,4…);
  ```

- **strtolower(*cadena*)**. Convierte toda la cadena en minúsculas.
- **strtoupper(*cadena*).** Convierte toda la cadena a mayúsculas.
- **ucfirst(*cadena*)**. Convierte la primera letra de la cadena en mayúscula.
- **ucwords(*cadena*)**. Convierte la primera letra de cada palabra de la cadena en mayúscula.
- **str_replace(*caracter_a_buscar,carácter_a_reemplazar,cadena*).** Reemplaza un carácter por otro en una cadena.

Operadores complementarios para las sentencias *WHERE*.

- **=,** igualdad.
- **< , >,** mayor que o menor que.
- **<= , >=**, menor o igual y mayor que o igual que.
- **!=**, diferente de.
- **LIKE**, parecido a, en este complemento se puede hacer uso del operador % que quiere decir, cualquier cosa, ya sea antes o después de la palabra escrita.

BIBLIOGRAFIA

Alba León, H. C. (2008). Introducción a la seguridad en cómputo [Software de cómputo]. México: Centro Mascarones, UNAM.

Aparicio Arista, R. E. & Rosas Bernal, R. (2007) Notas para el curso Lenguaje de Programación JAVA [Software de cómputo]. México: Centro Mascarones, UNAM.

Danesse, F. *Firewall.* Recuperado el 22 de marzo de 2010, de http://sites.google.com/site/flaviodanesse/firewall

Espinoza Altamirano, S. (2008). Editores para la creación de páginas web [Software de cómputo]. México: Centro Mascarones, UNAM.

Gallego Vázquez, J. A. (2003). *Desarrollo Web con PHP y MySQL*. Madrid: Editorial Anaya Multimedia.

Gonzáles Trejo, C. & Cendejas Cervantes, N. (2008). Instalación y administración de LINUX [Software de cómputo]. México: Centro Mascarones, UNAM.

Gonzáles Trejo, C. & Cendejas Cervantes, N. (2008). Sistema Operativo LINUX [Software de cómputo]. México: Centro Mascarones, UNAM.

Hernández Orozco, J.U. (2008). Interacción de WWW con Bases de Datos [Software de cómputo]. México: Centro Mascarones, UNAM.

Martínez Rico, O. (2007). *Guías y Textos de Cómputo. Desarrollo de aplicaciones web con Java JSPs y SERVLETs.* (1ª. ed.). México: UNAM.

Rangel Álvarez, A. (2009). Desarrollo de aplicaciones web con PHP y PosgreSQL [Software de cómputo]. México: Centro Mascarones, UNAM.

Rangel Álvarez, A. (2008). Programación con PHP [Software de cómputo]. México: Centro Mascarones, UNAM.

Román Zamitiz, C. A. *Programación Orientada a Objetos en Java*. Recuperado el 15 de mayo de 2007, de http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/Index.htm.