UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"INFORME DE UN SISTEMA DE CONTROL DE INFORMACIÓN IMPLEMENTADO CON SOFTWARE LIBRE"

T R A B A J O E S C R I T O

EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL QUE PARA OBTENER EL TÍTULO DE

I N G E N I E R O E N C O M P U T A C I Ó N

P R E S E N T A :

VILLALOBOS GONZÁLEZ BRENDA PAULINA

A S E S O R A :

ING. SILVIA VEGA MUYTOY

MÉXICO 2006

Dedicatorias:

A Dios:

Respondes cuando te llamo, le infundes a mi alma valor y aunque camine en medio de angustias tú me das vida.

Gracias por darme tanto amor.

A mi mamita (María de Jesús Lina González Borja):

Quien es la luz en mi camino, la confianza en mi andar, el consuelo en mi sufrir, mi alegría al despertar, el mejor ejemplo de madre, amiga, mujer pero sobre todo la inspiración de mi vida.

Gracias por prepararme para enfrentar la vida, de pequeña me inculcaste los valores del respeto, la gratitud, la honradez, la tolerancia, la honestidad, el trabajo, la responsabilidad, el perdón, la generosidad; ahora quiero que tengas la plena seguridad que me convertiste en una mujer de bien, capaz de enfrentarme a la adversidad. De manera escrita hago el compromiso con Dios y contigo que cada día seré una mejor persona.

Mami, tu vida dedicada a mi no es en vano soy muy feliz. Este triunfo es de las dos.

Mi vida te la dedico a ti. Por siempre te amaré.

A mi hermanita Moni:

Siempre has sido el mejor ejemplo ha seguir, tu personalidad influyo mucho en mi, eres la mejor hermana que Dios pudo darme. Te amo con toda el alma. Nuestro amor es infinito.

A mi hermanita Karlita:

Eres el regalo que Dios me envió para alivianar mi soledad. Hermanis desde el día que llegaste a mi vida has sido mi mejor amiga, te amo mucho, me siento muy orgullosa de ti. Nuestro amor es infinito.

A mis abuelitos (Raúl González Herrera y Esther Borja Pita)

Gracias su infinito amor y cuidados. Abuelitos este logro también es de ustedes, porque ustedes fueron mis guías en la infancia. A los amo mucho.

A mi papá (Gilberto Villalobos Gómez):

Papi todos los días de mi vida has estado en mi corazón y pensamientos, me siento muy orgulloso de ser tu "chatita". Te amo con toda mi alma.

A mi amor:

John gracias por tu infinita paciencia, apoyo y tu inquebrantable confianza en mi.

A ti debo gran parte de este logro porque hubo muchos momentos que sentí que no podía o veía interminable el trabajo, pero tu siempre tuviste una palabra de aliento, de confianza y ánimo para mi. En ocasiones no pudiste evitar que tropezara pero siempre me has levantado.

Gracias por todo lo que hemos vivido juntos, por lo que me haces sentir, por tu amor, por tus mil detalles, por tus consejos, por la persona que soy desde que te conocí, John este logro se suma a nuestra historia juntos.

Te amo, esta Cuchu no seria lo mismo sin ti.

Brenda Paulina Villalobos González

Agradecimientos

Agradezco a todas aquellas personas que han estado conmigo, porque ustedes han sido una gran influencia en mi personalidad y un apoyo muy grande en mi vida. Gracias, por darme tanto.

A mis grandes amigas Liz y Karina:

Las quiero mucho, gracias por tantos momentos felices juntas, por su apoyo, por cuidarme tanto, ustedes han sido una gran influencia para mi. Me siento muy orgullosa de ser su amiga.

A mi amiga-hermana Lilia:

Llevamos mucho tiempo persiguiendo esta meta juntas además eres un testigo fiel de las dificultades y triunfos que he logrado. Amiguis eres muy importante para mi, en ti encontré a un ser extraordinario, comprensivo, y muy amoroso. Dios puso en mi camino a una amiga maravillosa, Lilia ¡gracias!

A mi gran amigo Pedro:

Eres sin duda un gran ángel que Dios puso en mi camino. Me siento muy afortunada por tener tu amistad.. Pedro ¡gracias! por todo lo que me enseñaste, por cuidarme, por procurarme, por tener fe ciega en mi, no te voy a defraudar.

A Martha Díaz:

Por abrirme las puertas de tu casa, por ser tan cariñosa conmigo desde el primer momento en que te conocí, eres un gran ejemplo de mujer y madre. Gracias.

A todos mis amigos de la bandota y los que no lo son también por ser simplemente los mejores:

Mónica Juárez Monroy , Erika Arreguín Murillo, Samantha Pimentel García, Ruth Mancilla González, Juana Jacqueline Márquez Espino, , Janeth Cruz Ángel , Erubiel Ruiz Mosqueda, Juan Carlos Jarquín Ortega, Mario Alberto Buendía Hernández, Hugo Valle Guzmán, Eduardo González Martinez, Felipe Pérez García, Armando Blanco García, Mardol Benavides Ponce, Alfredo Osorio Martínez, Luis Alexander Reyes Pedro Bautista Fernández, Cesar Cárdenas Desales, David Escutia Lazcano, Omar Flores Cano, Juan Carlos Yescas Quiroz. Abraham Corona Gallegos y Alberto Reyes Jiménez.

A mi asesora Ing. Silvia Vega Muytoy:

Por su enseñanza y dedicación para lograr este objetivo. ¡Muchas gracias!

A mis amigos y profesores a quienes les debo mi formación profesional:

Ing. Octavio Mejía Sandoval, Ing. Martín Hernández Hernández, Ing. Narciso Hernández, Maestro. Ernesto Peñaloza Romero y Lic. Primo García.

A mi Alma Mater por darme las herramientas para enfrentarme a la vida profesional.

Brenda Paulina Villalobos González

ÍNDICE

Objetivo----------8
Introducción----------9
Capitulo 1- informe general del diplomado----------12
1.1 Sistema Operativo Linux----------12
1.1.1 Breve historia de Linux----------12
1.1.2 Características----------12
1.1.3 Plataformas y distribuciones más comunes----------13
1.1.4 Software de aplicación----------14
1.1.5 Requerimientos de instalación----------14
1.1.6 Imágenes ISO----------14
1.1.7 Algunos shell de Linux----------15
1.1.8 Comandos y utilerías básicas----------15
1.1.9 Metacaracteres----------16
1.1.10 Privilegios y permisos----------18
1.1.11 Estructura de archivos y directorios----------19
1.1.12 Editores de texto----------20
1.1.13 Ligas----------21

1.2 Instalación y administración de Linux----------23
1.2.1 Tareas administrativas comunes----------23
1.2.2 Conocimientos del administrador----------23
1.2.3 Instalación----------24
1.2.4 Descripción de los directorios----------26
1.2.5 Mantenimiento y claves de usuarios----------27
1.2.6 Comandos básicos pala la instalación de Linux Slackware----------27

1.3 Editores para la creación de páginas Web----------29
1.3.1 Etiquetas HTML----------29
1.3.2 Cabeceras----------30
1.3.3 Párrafos----------30
1.3.4 Listas ----------30
1.3.5 Texto preformateado----------31
1.3.6 Vínculos HTML----------31
1.3.7 Formas GET y POST----------32
1.3.8 Tablas HTML----------32

1.4 Administración de servidores WWW----------33
1.4.1 Definición de un servidor WWW----------33
1.4.2 Funcionamiento de HTTP----------33
1.4.3 ¿Por qué Apache?----------34
1.4.4 Compilar e instalar Apache----------35
1.4.5 Iniciar, detener y reiniciar el servidor----------37
1.4.6 Directivas de configuración----------38

1.5 Programación con PHP----------41
1.5.1 Lenguajes de cliente o lenguajes de servidor----------42
1.5.2 ¿Cómo funcionan las paginas PHP? ----------42
1.5.3 ¿Qué se necesita para poder programar con PHP? ----------43

1.5.4 ¿Qué se puede hacer con PHP? ----43
1.5.5 Conceptos básicos----43
1.5.6 Variables y tipos de datos----44
1.5.7 Expresiones----44
1.5.8 Sentencias----46
1.5.9 Funciones----48
1.5.10 Pasando argumentos----49
1.5.11 Arrays----49
1.5.12 Manejo de archivos----50
1.5.13 Pasaje de variables a través de PHP----52
1.5.14 Cookies----52

1.6 Interacción de WWW con bases de datos----54
1.6.1 Tipos de bases de datos----54
1.6.2 Introducción a SQL----54
1.6.3 Creación de la base de datos----55
1.6.4 Borrado de la base de datos----55
1.6.5 Crear una tabla----55
1.6.6 Borrar una tabla----55
1.6.7 Alta de un registro----56
1.6.8 Consulta sobre los registros existentes en una tabla----56
1.6.9 Ordenando resultados----57
1.6.10 Borrando un registro----57
1.6.11 Edición de un registro----57
1.6.12 ¿Qué es MySQL? ----58

1.7 Introducción a la seguridad en cómputo----61
1.7.1 Historia de la seguridad----61
1.7.2 Conceptos de la seguridad----61
1.7.3 Control de acceso----63
1.7.4 Super-usuarios: ¿Qué son y para qué sirven? ----64
1.7.5 Administración básica de la seguridad----64
1.7.6 Seguridad en Red----66
1.7.7 Implicaciones de la seguridad----66
1.7.8 Mecanismos de confianza----67
1.7.9 Políticas de seguridad----67

1.8 Desarrollo de aplicaciones de PostgreSQL y PHP----69
1.8.1 Lenguajes con soporte a la POO----69
1.8.2 ¿Qué es un objeto? ----69
1.8.3 ¿Qué es un clase? ----69
1.8.4 Los tres principios de la programación orientada a objetos----70
1.8.5 Otras características de la POO----71
1.8.6 Objetos, métodos y constructores----71
1.8.7 Atributos y métodos static----72
1.8.8 Visibilidad----72
1.8.9 Interfaces----73
1.8.10 Excepciones----73
1.8.11 Templates ----73

Capitulo 2- Proyecto de un sistema de control de información por intranet----------76
2.1 Objetivo----------76
2.2 Justificación del Proyecto----------76
2.3 Descripción del proyecto----------77
2.4 Software utilizado para la implementación----------77
2.5 Características del cliente----------78
2.6 Base de datos----------78
2.6.1 Diagrama Entidad Relación de la base de datos----------79
2.6.2 Diccionario datos de la base de datos----------79
2.7 Especificación de los módulos del proyecto----------80
2.7.1 Menú principal----------81
2.7.2 Módulo Redactar Reporte----------84
2.7.2.1 Búsquedas de reportes----------85
2.7.2.2 Redactar de reporte----------88
2.7.2.3 Reportes----------90
2.7.3 Modulo validar reporte----------93
2.7.3.1 Búsqueda de reportes----------94
2.7.3.2 Validar reporte----------98
2.7.3.3 Reportes----------99

Conclusión----------102
Glosario----------103
Bibliografía----------107

OBJETIVO GENERAL

Implementar una mixtura entre dos alternativas: el aspecto teórico y el aspecto práctico, para poder exponer de una forma completa todo lo visto en el diplomado séptima edición: "Desarrollo de Sistemas Implementados con Software libre: Linux".

OBJETIVOS ESPECÍFICOS

- Identificar los métodos de operación y administración del sistema operativo, utilizando los programas de manejo de usuarios, y llevando a cabo la instalación del dispositivo de respaldo, configuración de la red e instalación del software.
- Instalar el sistema operativo Linux en PCs y configura la tarjeta, particiones del disco duro, así mismo manejar el ambiente de usuario.
- Elaborar paginas WWW, apoyado con HTML.
- Identificar el procedimiento para instalar, configurar y administrar su propio servidor WWW en un servidor de plataforma Linux.
- Crear aplicaciones dinámicas e interactivas para el Web utilizando el lenguaje PHP.
- Desarrollar una aplicación de base de datos que funcione a través del WWW, empleando herramientas de software libre.
- Reconocer la importancia de la seguridad e identificar los elementos que permitan proteger el sistema de información.

INTRODUCCIÓN

Es importante desarrollar software al alcance de cualquier usuario, para ello se adoptará un enfoque de desarrollo en una plataforma de software libre, como es el caso de Linux, ahorrándose así, licencias innecesarias que haya que pagar. En caso de optar por otra plataforma, este sistema implementaría un servidor con Apache, como servidor de páginas Web, MySQL como servidor de base de datos y como lenguaje de programación PHP.

México debe de estar a la vanguardia ante los cambios tan fuertes a nivel Nacional e Internacional, por lo tanto es importante desarrollar aplicaciones que sean funcionales, prácticas y al alcance de los usuarios.

Por las anteriores razones se requieren en el país, profesionales que puedan dar respuesta a éstas necesidades, es por eso que es necesario cuidar de todos aquellos aspectos y elementos que intervengan para poder llevar a una solución exitosa

El Sistema Operativo Linux se ha convertido en una de las más importantes distribuciones de Unix, brindando a la PC toda la potencia y flexibilidad de una estación de trabajo Unix, además de un completo conjunto de aplicaciones de Internet y una interfaz de escritorio altamente funcional.

En cuanto a Internet se refiere, Linux se ha convertido en una plataforma para aplicaciones muy potentes. Con Linux, no sólo puede utilizar Internet, sino, también, formar parte de ella, creando sus propios sitios Web, FTP o Gopher. Otros usuarios, incluso varios al mismo tiempo, pueden acceder a sus sistemas Linux, utilizando diferentes servicios: También puede utilizar potentes clientes de Unix, Gnome y KDE para correo y noticias. Los sistemas Linux no se limitan a Internet. Puede utilizar Linux en cualquier intranet local, configurando un sitio Web o FTP.

Para poder implantar un sistema es preciso estar actualizado, manejar las herramientas adecuadas y conocer el tema por tal motivo es que el diplomado de desarrollo e implementación de sistemas con software libre me interesó porque da a conocer nuevas herramientas informáticas que permiten desarrollar sistemas con software libre.

A través del presente informe, daré un resumen de aquellos aspectos más importantes que a mi punto de vista me sirvieron para poder desarrollar un proyecto de un sistema de control de información.

El informe esta dividido en dos capítulos. En el primer capítulo se dará un informe general del diplomado; a su vez este capítulo esta dividido en ocho subcapítulos, cada uno aborda cada módulo del diplomado. El subcapítulo uno es una visión general de el sistema operativo Linux. El subcapítulo dos son los aspectos a considerar en la administración de un sistema Linux. El tercer subcapítulo mostrará algunas etiquetas html para la elaboración de interfaces de usuarios. Apache será detallado en el subcapítulo 4, su instalación, configuración y algunas directivas. El lenguaje propuesto en el diplomado para programar sistemas es PHP, el cual es introducido en el subcapítulo 5. El subcapítulo 6 trata de la interacción de www con las bases de datos, MySQL como sistema gestionado de base de datos. La introducción a la seguridad en los sistemas es abordada en el subcapítulo 7. En el subcapítulo 8, estudiamos otro sistema gestionador de base de datos

PostgreSQL, que beneficios tiene frente a MySQL, PHP5 y su programación orientada a objetos.

El capítulo dos aborda la construcción de un Sistema de Control de Información para una intranet implementada con software libre. Apache como Servidor Web, lenguaje de programación PHP y como sistema gestionador de base de datos MySQL.

CAPÍTULO 1

INFORME GENERAL DEL DIPLOMADO

CAPÍTULO 1. INFORME GENERAL DEL DIPLOMADO

1.1 SISTEMA OPERATIVO LINUX

Linux es un sistema operativo para equipos personales y estaciones de trabajo que incorpora ahora una interfaz gráfica de usuario (GUI), como Windows y Mac (aunque más estable). Fue desarrollado a principios de la década de los noventa por Linus Torvalds, junto con otros programadores de todo el mundo. Dada su condición de sistema operativo, Linux desempeña las mismas funciones que Unix, Mac, Windows y Windows NT. Sin embargo se caracteriza por su flexibilidad y potencia. Por estar basado en UNIX ofrece potentes características de red, como por ejemplo, el soporte para Internet, intranets, redes Windows y redes AppleTalk. Por lo general, las distribuciones de Linux integran servidores de Internet rápidos, eficientes y estables, como servidores Web, FTP así como servidores de nombres de dominio, Proxy, de noticias, de correo y servidores de indexación. Es decir ofrece todo lo necesario para instalar, proporcionar soporte y mantener una red completamente funcional.

1.1.1 BREVE HISTORIA DE LINUX

Dado que se trata de una versión de UNIX, la historia de Linux comienza naturalmente con UNIX. Comienza a finales del sesenta, cuando se realizó un esfuerzo conjunto por desarrollar nuevas técnicas de sistema operativo. En 1968, un consorcio de investigadores llevó a cabo un proyecto especial de investigación de sistemas operativos denominado MULTICS. Este incorporaba numerosos conceptos sobre la función, multitarea, administración de archivos e interacción de usuarios. En 1969, Ken Thompson, Dennis Ritchie y algunos investigadores desarrollaron el sistema operativo UNIX el cual esta basado en MULTICS.

En un principio, Linux fue diseñado específicamente, para equipos personales basados en Intel. Linux nació como el proyecto personal de un estudiante de informática llamado Linus Torvalds, en la Universidad de Helsinki. Los estudiantes utilizaban un programa denominado Minix, que destacaba diferentes características de UNIX. La intención de Linus era crear una versión efectiva de UNIX para PC que pudiesen utilizar los usuarios de Minix. La llamo Linux y, en 1991, Linus lanzó la versión 0.11. Linux se distribuyó ampliamente a través de Internet y, en los años siguientes, otros programadores lo pulieron y aportaron nuevas ideas, incorporando la mayor parte de las aplicaciones y características que se pueden encontrar actualmente en los sistemas UNIX estándares.

1.1.2 CARACTERÍSTICAS

A diferencia de UNIX, Linux puede dividirse generalmente, en tres componentes principales: El **kernel**, el **entorno** y la **estructura de archivos**. El **Kernel**.- Es el programa base que ejecuta programas y administra los dispositivos de hardware, por ejemplo los discos, y las impresoras. En un sentido amplio, se denomina Kernel al núcleo del sistema operativo, y el **shell** uno de los principales programas de soporte. El **Entorno**.- proporciona una interfaz para el usuario; recibe comandos del

usuario y los envía al kernel para que sean ejecutados. La **Estructura de Archivos** organiza el modo en que se almacenan los archivos en un dispositivo de almacenamiento por ejemplo, un disco. Cada directorio debe contener un número de subdirectorios que, a su vez, contengan archivos. Juntos, el kernel, el entorno y la estructura de archivos forman la estructura básica del sistema operativo. Con estos tres componentes, se pueden ejecutar programas, administrar archivos e interactuar con el sistema.

Linux se distribuye sin costo alguno con licencia GNU, facilitando el aprovechamiento de las computadoras personales actuales y poniendo a disposición del usuario los archivos del sistema con todo el código fuente.

Linux es un sistema operativo multiusuario y multitarea de dificultad evidente, la computadora que utilice este sistema operativo tendrá que atender a una gran variedad de procesos requeridos por los distintos usuarios que estén en ese sistema y al mismo tiempo, deberá planificar su control.

Linux se desarrolló como un esfuerzo conjunto a través de Internet, de modo que ninguna empresa ni institución controla Linux. El software desarrollado para Linux pone de manifiesto esta condición. Se producen nuevos avances, a menudo, cuando usuarios de Linux deciden trabajar juntos en un proyecto. Una vez terminado, el software se coloca en un sitio de Internet y cualquier usuario de Linux puede acceder al sitio y descargar el software.

La mayor parte del software para Linux se desarrolla como código fuente abierto. Es decir, que el código fuente de una aplicación se distribuye gratuitamente junto con la aplicación. Los programadores pueden contribuir al desarrollo del software a través de Internet, modificando y corrigiendo el código fuente. Su código fuente se incluye en todas las distribuciones y se puede conseguir gratuitamente en Internet.

El software de código fuente abierto está protegido por licencias públicas; de este modo se evita que las empresas comerciales tomen el control del software de código fuente abierto añadiendo unas cuantas modificaciones propias, protegiendo legalmente estos cambios y vendiendo el software como si fuera su propio producto. La licencia pública más conocida es la GNU Public License concedida por Free Software Foundation. Esta es la autorización en la que se ampara la distribución de Linux. La GNU Public License conserva los derechos de autor, garantizando licencias libres sobre el software con la condición de que tanto el software como cualquier modificación introducida se pueden obtener siempre libremente.

1.1.3 PLATAFORMAS Y DISTRIBUCIONES MÁS COMUNES

A pesar de que hay una única versión estándar de Linux, en realidad, hay varias ediciones. Diferentes empresas y grupos han empaquetado Linux y el software de Linux con ligeras diferencias y, a continuación, cada empresa o grupo han lanzado el paquete de Linux, normalmente en un CD-ROM. Las ediciones más recientes pueden incluir versiones actualizadas de programas o software nuevo.

El kernel de Linux es, distribuido centralmente a través de www.kernel.org. Todas las

distribuciones utilizan el mismo Kernel, aunque puede estar configurado de modo diferente.

A continuación se listan las distribuciones más populares

- Red Hat
- Fedora
- Peanut Linux
- Suse
- Debian
- Mandrake
- Slackware
- Gentoo
- Yellow Dog
- United Linux
- Darwin
- Turbo Linux
- Free BSD
- Ubuntu

1.1.4 SOFTWARE DE APLICACIÓN

OpenOffice.Org
Navegadores: Mozilla, FireFox, Galeon, Konqueror
Lenguajes de programación: php, Java, Perl, Shell, Pascal, Kylix, Ada, Python, C#
Emulador: VMware, Dosemu.

1.1.5 REQUERIMIENTOS DE INSTALACIÓN

Antes de instalar Linux, se debe asegurar de que la computadora cumpla los requisitos mínimos de instalación[1].

- Procesador.- Un procesador 386.
- Memoria.- 4 MB de memoria RAM.
- Requerimiento de disco.- 40 MB de disco duro pero si se quiere instalar todo se requiere de 400 MB.
- Unidades de disquete o CD ROM.

Linux soporta una gran variedad de dispositivos como son ratones, impresoras, escáners, tarjetas de red, etc. Sin embargo no se requieren ninguno de estos dispositivos durante la instalación del sistema.

1.1.6 IMÁGENES ISO

Imagen ISO es un archivo donde se almacena una copia o imagen exacta de un sistema de ficheros, normalmente un disco compacto (como un CD o un DVD). Se rige por el estándar ISO 9660 de la Organización Internacional para la Estandarización.

Por esta razón es muy utilizado a la hora de distribuir sistemas de software libre Linux, BSD u otros por Internet, para evitar en la transferencia la pérdida de cualquier información. Aunque la ISO 9660 lo especifica como formato de sólo lectura es posible modificarlos con algunos programas.

[1] De Petersen Richard, Manual de referencia Linux, (2000, p 27)

Filosofía de Unix

"Lo pequeño es mejor".

No existen utilerías que realicen múltiples tareas

1.1.7 ALGUNOS SHELL DE LINUX

LINUX ofrece distintos procesadores de órdenes o shells, la tabla 1.1 muestra los más comunes.

Tabla 1.1 Shells comunes en Linux

Programa Shell	Nombre
Sh	Bourne shell
Cts.	C shell avanzado
Ash	Reducido shell
Zsh	Z shell
Bash	Bourne Again Shell
Csh	C shell
Pdksh	Korn shell de dominio público

1.1.8 COMANDOS Y UTILERÍAS BÁSICAS

Tabla 1.2 Comandos y utilidades principales de Linux

Comando	*Descripción*
Pwd	Imprime el PATH completo de donde se esta ubicado.
Date	Informa sobre el día de la semana, el día del mes, la hora con los segundos de la zona horaria y el año.
Bc	Calculadora.
Quit	Abandona la utilidad calculador.
Medir	Crea directorios.
Touch	Se usa para crear archivos.
Ls	Lista directorios.
Echo	También se puede usar para enumerar el contenido de los directorios. Imprime todos los nombres de archivo coincidentes en orden alfabético, pero no da formato al listado en columnas.
Cat	Imprime el contenido de los archivos en la consola o ventana del terminal. Este comando sirve muy bien para imprimir archivos cortos en la pantalla, ya que los archivos largos irán demasiado rápidos para poder leerlos.
less	Esta diseñado para permitir que el usuario se desplace por múltiples páginas de un archivo, deja imprimir el contenido del archivo en la pantalla al final de la misma, en vez de al final de un archivo, también se puede usar con operadores de redireccionamiento de entrada y salida.

Comando	*Descripción*
More	Al igual que less se emplea para leer archivos de texto interactivamente, sólo que este se termina automáticamente cuando llegue al final del archivo. El comando more sólo le permite desplazarse hacia adelante por el archivo, mientras que el comando less le permite moverse hacia adelante y hacia atrás.
cp	Cambia de directorio de trabajo, si no se especifica ningún directorio, Linux retornara a su directorio inicial.
mv	Se usa para mover o cambiar el nombre de los archivos y los directorios.
ln	Crea vínculos simbólicos.
rm	Elimina archivos y directorios.
rmdir	Elimina directorios.
find	Busca archivos.
locate	Localiza rápidamente archivos o directorios en su sistema.
whereis	Se usa para enumerar las ubicaciones de los binarios de programa, archivos relacionados y páginas de manual.
which	Es más rudimentario que el comando whereis, pero también esta diseñado para ayudar a localizar un archivo o una aplicación. Este comando sencillamente comprueba si el programa especificado se encuentra en alguna parte de la ruta.
Grez	Devuelve coincidencias de palabras como patrones en la línea de comandos.
Strings	Para extraer cadenas de archivos binarios.
who	Muestra información del nombre del usuario.
man	Muestra el manual de los comandos.

1.1.9 METACARACTERES

Los metacaracteres son caracteres que tienen un significado en el shell.

Cuando se utiliza algún metacaracter, los comandos no lo reciben, sino que el shell lo reemplaza por lo que corresponda, y le pasa al comando ejecutado el resultado de ese reemplazo.

Eso es lo que se entiende por interpretar: reemplazar el carácter por otro carácter o por una cadena de caracteres, según corresponda. Se listan en la Tabla 1.3.

Metacaracteres relacionados con comandos

Ejecutar un comando es tan sencillo como escribir el comando y apretar ENTER. Sin embargo, utilizando algunos de los metacaracteres de shell se pueden combinar los comandos entre sí, y lograr resultados mucho más importantes. Los comandos relacionados con comandos se listan en la tabla 1.4.

Tabla 1.3 Lista de Metacaracteres.

Metacaracter	***Definición***
*	Cuando el shell encuentra un *, lo reemplaza por una lista de los archivos que concuerdan con la expresión indicada. En el caso de que no hubiera ningún archivo que concuerde con la expresión, generalmente, mostrará la expresión que se haya escrito.
?	Al encontrar un ? el shell lo reemplaza por cualquier otro carácter. Es decir que la expresión que se escriba se reemplazará por todos los archivos que en esa posición tengan cualquier carácter, y en el resto de la cadena tengan lo que se ha escrito. Al igual que con el *, si ningún archivo concuerda con el patrón, generalmente, muestra la misma expresión que se a escrito.
[ ]	Encerrados por los corchetes, se puede escribir un rango de caracteres con los cuales se quiere que el shell concuerde. Se puede además especificar un rango de caracteres, con un guión en el medio. Por ejemplo, a-z (letras minúsculas), 0-9 (números), etc. y combinarlos con caracteres individuales siempre que no sea ambigua la interpretación. (Considerar la concordancia con el carácter -). Por ejemplo, se puede querer sólo los archivos que comienzan con números seguidos de un -, en ese caso se escribiría ls [0-9]-*} o ls [0-9][0-9]-*
^	Cuando al comienzo de la cadena que está encerrada por los corchetes se encuentra el carácter ^, se esta indicando que debe concordar los caracteres que no se encuentran en el rango.

Tabla 1.4 Metacaracteres relacionados con comandos.

Metacaracter	***Definición***
;	El ; es un separador de comandos, permite ejecutar un comando a continuación de otro, equivalente a lo que sucedería si ejecutara primero uno, y al terminar se ejecutara el siguiente.
( )	Los paréntesis sirven para encerrar grupos de comandos, y tratarlos como si fueran uno solo.
&	El & manda el comando a background, esto quiere decir, que devuelve la línea de comandos inmediatamente después de apretar Enter, mientras el comando sigue ejecutándose en segundo plano.

1.1.10 PRIVILEGIOS Y PERMISOS

Los permisos de Linux, es una parte fundamental en su aprendizaje, ya que involucra el acceso, la seguridad, la accesibilidad y la comodidad de los usuarios para ejecutar sus tareas de acuerdo al nivel de acceso que éste posee en el sistema. Es importante por esto aprender como funciona y cual es la importancia del comando chmod, así como saber también las posibles consecuencias de un mal uso o de una no-aplicación de este último.

El sistema proporciona un mecanismo conocido como permisos para proteger ficheros de usuarios del sistema, de la manipulación de otros usuarios, ya que Linux como se sabe es multiusuario. Los permisos están divididos en tres tipos:

1.- permisos de lectura, representado por la letra "r".
2.- permisos de escritura, representado por la letra "w".
3.- permisos de ejecución, representado por la letra "x".

A su vez estos permisos pueden ser fijados para tres clases de usuarios:
1.- dueño del archivo: u.
2.- grupo al que pertenece el archivo: g.
3.- el resto de usuarios: o.

El comando chmod es el que te va a permitir manipular todos los permisos a tu gusto, consta de 3 operadores, veamos cuales son:

1.- "+" para agregar permisos.
2.- "-" para quitar permisos.
3.- "=" para asignar permisos.

Por ejemplo, se puede establecer para asignar permisos de ejecución a un fichero: chmod +x fichero. En este caso estamos dándole permisos de ejecución, tanto para el usuario propietario, el grupo y para el resto de usuarios, pero ¿qué sucede, si solo se que el propietario del sistema tenga permisos de ejecución del fichero, y no todos los demás como lo acabo de hacer anteriormente?

Pues para se indica: $chmod u+x fichero

Análogamente se puede hacer para darle permisos al grupo o al resto con las letras g y o respectivamente.

Definiendo permisos con el sistema octal

Existe otro método para definir permisos. Este sistema se llama: "sistema octal". En este sistema los números representan permisos. Por ejemplo: 0001, 0100, 0400, 1000, etcétera.

Estos se basan en la suma de los 3 valores: los de lectura, escritura y ejecución
ejecución -> valor 1
escritura -> valor 2
lectura -> valor 4

La combinación de estos, da números del cero al siete, de esta manera:

0 = sin permisos.
1 = ejecución.
2 = escritura.
3 = escritura y ejecución.
4 = lectura.
5 = lectura y ejecución.
6 = lectura y escritura.
7 = lectura, escritura y ejecución.

Veamos un ejemplo:

$chmod 755 fichero

Esto quiere decir que al propietario le estamos dando los permisos de lectura, escritura y ejecución. Para el grupo los permisos de lectura y ejecución, igualmente para el resto de usuarios, ya que ambos tienen el permiso 5.

1.1.11 ESTRUCTURA DE ARCHIVOS Y DIRECTORIOS

Linux proporciona estructura y soporte para poder gestionar sus propios archivos. Esta estructura consiste en una serie jerárquica de niveles de directorio. Existe un directorio llamado *raíz*, el cual se divide en varios subdirectorios, que, a su vez, se pueden ramificar en subdirectorios.

A cada usuario se le asigna un subdirectorio propio, de tal manera que cuando se accede a esta dirección, se convierte en su directorio de trabajo actual.

Cuando se es superusuario se utilizará habitualmente los siguientes archivos y directorios (Tabla 1.5).

Tabla 1.5 Directorios del sistema de Linux

Directorio (/)	Contenido
/bin	Ficheros binarios o ejecutables
/dev	Dispositivos especiales
/etc	Información y programas
/sbin	Órdenes ejecutables sólo por el administrador
/home	Directorio de usuario
/lib	Librerías
/proa	Estructura virtual de archivos
/tmp	Archivos temporales
/usr	Archivos de configuración y programas usados por el sistema
/var	Históricos del sistemas
/boot	Información necesaria para el sistema de arranque
/cdrom	Manejadores para el CD-ROM

Directorio (/)	Contenido
/dev/console	Sistema de consola
/dev/ttyS	Acceso a puertos
/dev/cua	Acceso a puertos
/dev/hda	Primer disco duro
/dev/sda	Primer disco duro SCSI
/dev/lpo	Primer puerto paralelo
/dev/tty	Consolas virtuales
/dev/pty	Seudoterminales
/usr/x386	Sistemas X Window
/usr/bin	Archivos binarios o ejecutables
/usr/etc	Información y programas
/usr/incluye	Archivos para compilar C
/usr/man	Pagina manual
/usr/src	Código fuente
/usr/src/Linux	Código fuente del núcleo
/var/adm	Archivos de administración
/var/spool	Archivos spool
/usr/x11/bin	Ejecutables X Windows

1.1.12 EDITORES DE TEXTO

Los editores de texto son importantes para los usuarios de Linux. Para utilizar Linux de manera eficiente y productiva se deberá elegir el editor de textos que mejor se adapte a nuestras necesidades y preferencias.

Los editores de texto que se usan para el procesamiento de textos suelen admitir el movimiento del cursor a través de un archivo y están orientados a la pantalla.

También se puede llevar a cabo la edición de textos con programas no interactivos como los filtros de texto o los editores de flujos.

Algunos de los procesadores de texto comerciales para Linux son:

- WordPerfect for Linux
- Applix Words
- StartOffice Star Writer

Los editores de texto son fáciles de utilizar, y suelen ser las herramientas preferidas de los programadores, escritores y eventuales usuarios. Emacs esta entre los más conocidos.

VI probablemente sea de los editores más usados en el mundo de UNIX. Es muy útil aprender al menos los fundamentos de la edición con VI, ya que se puede estar seguro de que algunas de sus versiones estará instalada en cualquier máquina.

Vi es un editor de texto. Hay una versión mejorada que se llama Vim, pero Vi es un editor de texto que se encuentra en (casi) todo sistema de tipo Unix, de forma que conocer rudimentos de Vi

es una salvaguarda ante operaciones de emergencia en diversos sistemas operativos.

Editar

Para editar un archivo de texto (digamos ma.txt) se debe hacer desde el editor de comandos.

Vi es un editor con dos modos: edición y comandos. En el modo de edición el texto que ingrese será agregado al texto, en modo de comandos las teclas que oprima pueden representar algún comando de vi. Cuando comience a editar un texto estará en modo para dar comandos el comando para salir es: seguido de q y ENTER --con ese comando saldrá sino ha hecho cambios al archivo o los cambios ya están salvados, para salir ignorando cambios: q! seguido de ENTER.

Puede insertar texto (pasar a modo edición) con varias teclas, la tabla 1.6 lista algunos de los comandos para utilizar el editor:

Tabla 1.6 Comandos del editor Vi

Comando	Definición
I	Inserta texto antes del carácter sobre el que está el cursor.
A	Inserta texto después del carácter sobre el que está el cursor.
I	Inserta texto al comienzo de la línea en la que está el cursor.
A	Inserta texto al final de la línea en la que está el cursor.
O	Abre espacio para una nueva línea después de la línea en la que está el cursor y permite insertar texto en la nueva línea.
O	Análogo al anterior, pero abre espacio en la línea anterior.

Para pasar de modo edición a modo de comandos se emplea la tecla ESC, para desplazarse sobre el archivo puede emplear las flechas, PgUp, PgDn, también se pueden utilizar las teclas j (abajo), k (arriba), h (izquierda) y l (derecha).

Para consultar otro comando, ya sea del editor vi o de cualquier tecleando: $ man comando se puede obtener ayuda, por ejemplo: man vi

El editor pico forma parte del paquete del software del programa de correo electrónico. Pese a que adolece de muchas de las características avanzadas que se encuentran en vi, es un editor fácil de usar. Este programa es compacto, fiable y eficiente. Soporta incluso el ratón, cuando se usa con la opción de línea de comandos -m.

1.1.13 LIGAS

Ligas suaves

Ligas suaves son punteros a programas, archivos o directorios en cualquier ubicación (parecido a los accesos directos de windows).

Si el programa original, archivo o directorio es renombrado, movido o borrado la liga suave se corta. Las ligas se muestran ls –F, se puede ver que es liga suave porque termina con @. Para crear una liga suave llamada myfilelink.txt que se direcciona a un archivo llamado myfile.txt, usa:

ln –s myfile.txt myfilelink.txt.

Ligas duras

Punteros a programas y archivos pero no a directorios, si el programa original, archivo o directorio es renombrado, movido o borrado la liga dura no se rompe.

1.2 INSTALACIÓN Y ADMINISTRACIÓN DE LINUX

El administrador es aquella persona responsable de configurar, mantener y actualizar el sistema o conjunto de sistemas que forman una red, cuidando el funcionamiento del software, hardware y periféricos.

La importancia del administrador radica en proporcionar un ambiente seguro, confiable y eficiente, además de brindar funcionamiento confiable del sistema. Pero la responsabilidad del buen funcionamiento del sistema no va a depender únicamente de una persona, es decir se puede dividir el trabajo entre varios administradores, esto va a depender del tamaño del sistema.

1.2.1 TAREAS ADMINISTRATIVAS COMUNES

- Administrar usuarios.
- Instalación y configuración de dispositivos (Impresoras, discos, unidades de respaldo, etc).
- Instalación y configuración de software (Comercial, dominio público).
- Configuración de las interfaces de la red.
- Administración de los recursos (CPU, memoria, disco duro).
- Capacitar y asesorar usuarios.
- Asegurar los sistemas.
- Registrar cambios del sistema.
- Monitoreo del sistema.
- Detección de fallas.
- Auditoria e implantación de la seguridad del sistema.
- Mantener canales de comunicación.
- Establecer políticas de seguridad.

1.2.2 CONOCIMIENTOS DEL ADMINISTRADOR

- Técnicas de programación.
- Dominio al menos de un lenguaje de programación.
- Funcionamiento del sistema operativo.
- Técnicas de administración del sistema operativo.
- Conocimientos básicos de hardware y mantenimiento de dispositivos.
- Comprensión profunda sobre el redireccionamiento, tuberías, procesamiento en segundo plano, etc.
- Manejo de vi, pues es el común denominador entre los sistemas UNIX.
- Programación en shell.
- Planear las actividades(Absolutamente todas las actividades de administrador se planean).
- Guardar copias de seguridad (Jamás modificar sin respaldar previamente).
- Conocimientos de utilerías del sistema.
- Control de tareas.
- Respaldos.

Además de estos conocimientos el administrador debe de conocer las características, modelos, ubicación de los elementos del sistema.

1.2.3 INSTALACIÓN

Preparativos antes de comenzar la instalación:

1. Si se tiene una PC con la posibilidad de arrancar desde el cd se observará que al momento de arrancar se muestra un mensaje como el siguiente *"Hit del to enter setup"* o *"Press F1 to enter setup",* (según la computadora que se tenga), se tendrá que pulsar una u otra tecla para entrar en la bios, hay que tener cuidado con modificar algo que no se indique aquí, a continuación se desplegará un menú, se debe entrar en la opción que pone "bios features setup", y dentro de este menú en la opción "boot sequence" y hay que pulsar las teclas av/pag, re/pag hasta que este primero el CD-rom, después hay que salir guardando los cambios, como siguiente paso se introduce el cd, esperar a que cargue y ya se esta en el programa de instalación.

2. Para crear una nueva partición a partir de las existentes si el disco esta lleno: desde dos o Windows se deberá utilizar algún programa para particionar el disco, actualmente existen muchos de ellos que se pueden descargar.

3. Después de arrancar con el diskette (Nº 1) que se acaba de crear y saldrá el promt de la instalación esperando para que se introduzcan parámetros, no escribir nada y pulsar enter, entonces la instalación invitara a que se inserte el diskette de instalación (el Nº 2), después de hacer esto saldrá a la pantalla de login donde deberás poner root (superusuario, es decir el que mantiene el sistema y deniega los permisos en sistemas operativos Unix a los demás usuarios, pero Linux generalmente si se instalan en una PC particular el usuario y el root son la misma persona).

4. Particionar el disco duro:
   Arrancando fdisk: Cuando se ejecuta fdisk se debe especificar la unidad que se va a usar. Por defecto el programa tratara de abrir /dev/hda, pero en algunos casos esta no es la unidad correcta para usar. Sólo especifique el nombre de la unidad después de escribir fdisk en la línea de comando. Por ejemplo: **fdisk /dev/hdb**

   Esto le dirá a fdisk que despliegue el disco duro IDE esclavo primario. Se observa que no se especifica el número de la partición en la unidad.

   Para comenzar el particionado se utiliza el programa fdisk tecleando fdisk desde la línea de comandos; Este programa es mucho más potente que el fdisk de msdos aunque tiene el inconveniente de que se ejecutan las órdenes mediante las teclas que enumero:

   a *toggle a botable flag:* Poner o quitar la marca de arranque de una partición.
   d *delete partition:* Eliminar una partición.
   l *list know part. types:* Mostrar los tipos de particiones reconocidas.
   m *print this menu:* Mostrar la ayuda.
   n *add a new partition:* Crear una nueva partición.
   p *print the partition table:* Mostrar la tabla de particiones.

q *quit w/out saving changes:* Salir sin salvar los cambios.
t *change a partition's sysid:* Cambiar el tipo de partición.
u *change display/entry units:* Cambiar unidades entre bloques y sectores.
v *verify the partition table:* Comprobar la tabla de particiones.
w *write table and exit:* Salir y salvar.
x *extra funcionality:* Opciones extra (no te lo recomiendo a no ser que seas un experto).

Se debe crear una partición Linux swap (Memoria virtual) de al menos 16 Mb como mínimo y una partición Linux native (partición principal) de al menos 40 Mb (aunque se recomienda 200 para la principal y 32 para la virtual ya que en 40 Mb casi no se puede instalar casi nada).

Siempre es una buena idea crear la partición swap primero así se podrá especificar el tamaño exacto. También es una buena idea hacer particiones separadas para /, /home, y /usr. La gente le dirá muchas cosas acerca de como dividir su disco duro, pero lo que realmente importa es para que se necesite. Hay buenas razones para separar /, /home, y /usr. por ejemplo:

Los directorios Home siempre tendrán su propia partición y así poder actualizar la distribución sin tener que hacer un respaldo de los directorios home, /usr es donde va el software así que se podrá mantenerlo siempre aunque se actualice la distribución, el directorio raíz debería permanecer intacto, salvo los archivos modificados en el directorio /etc y el directorio home de root. También se podrá tener una partición para /var separada para que lo archivos log no llenen el directorio raíz, o para que el email spool tenga su propia partición. Finalmente la decisión va a depender de las necesidades que se tenga.

5. El programa de setup: Si se tiene 4Mb de ram o menos tendremos que activar la memoria virtual con el siguiente comando: mkswap /dev/hd** **** (los dos primeros asteriscos se sustituir por la posición de la partición, hda1 por ejemplo que indica que esta en el primer disco duro IDE la hd y la a indica que es el primer disco duro y el 1 que es la primera partición del disco duro, sustitúyelos por lo que sea el caso. Los siguientes asteriscos se sustituyen por el tamaño en Kb de la partición swap que se ha creado, cada Mb equivale a 1024Kb) y el comando swapon/dev/hd**

6. El siguiente paso es ejecutar el programa de setup tecleándolo desde la línea de comandos: que tendrá las siguientes opciones:

Help: Muestra la ayuda.
Keymap: Selección del tipo de teclado, generalmente "es" (español).
Quick: Sirve para cambiar entre la instalación silenciosa (Quick) o explicada (verbose), esto afecta a la forma en la que se mostraran las opciones durante la instalación.
Make tags: Los "tags" son los ficheros a partir de los que se indica cuales ficheros son imprescindibles y cuales son opcionales (por los cuales se pregunta al instalar). Esto suele ser innecesario a no ser que se instale en múltiples PC's.
Addswap: Sólo se deberá usar esta opción sino se ha hecho antes de entrar en la instalación.
Target: Se indica aquí la partición donde se va ha instalar el sistema (Linux native), si no esta formateada el sistema preguntara si se quiere formatear, se debe decirle que si.
Source: Aquí se debe indicar desde donde vas a instalar Slackware generalmente el CD.
Disk sets: En este apartado se debe indicar los paquetes a instalar para hacerlo más sencillo basta con que selecciones la serie A.
Install: Una vez que ya se haya pasado por las opciones necesarias para comenzar la instalación.

Configure: Tras realizar la instalación aquí se deberá configurar otros paquetes, como por ejemplo la instalación de LILO (el gestor de arranque de Linux que se instala en el master boot record).

7. Durante la instalación saldrá la pregunta sobre la creación de un diskete de arranque, que sirve para arrancar Linux sin utilizar el gestor de arranque. También pregunta sobre el sistema horario hay que seleccionar America/Mexico city. Se preguntara sobre el ratón, el modem y el CD-Rom y los puertos a los que se conectan.

8. Una vez finalizada la instalación *se pueden añadir nuevos paquetes* accediendo de nuevo a este programa tecleando setup desde la línea de comandos de Linux.

1.2.4 DESCRIPCIÓN DE LOS DIRECTORIOS

Tabla 2.1 Descripción de los directorios de Linux

Directorio	Descripción
/bin	Programas relacionados con el uso básico. Comandos del shell y programas como el *ls*.
/boot	LILO y archivos relacionados con el inicio
/dev	Archivos y bloques de dispositivos
/etc	Archivos de inicialización y configuraciones.
/home	Directorios Home de todos los usuarios excepto root
/lib	Librerías esenciales (como las librerías del sistema C y módulos del kernel).
/mnt	Punto de montaje genérico para agregar dispositivos
/opt	Programas opcionales. Slackware instala el KDE en este directorio.
/proa	Punto de montaje del Proc filesystem para interactuar con el kernel
/root	Directorio Home de root.
/sbin	Binarios del sistema. Programas ejecutados por root o en el inicio.
/tmp	Directorio temporal. Todos los usuarios tiene permiso de lectura+escritura en este directorio.
/usr	Programas relacionados con el usuario, como X11, netscape y *pine*.
/var	Archivos log del sistema, archivos de bloqueo, colas de impresión y colas de correo.

1.2.5 MANTENIMIENTO Y CLAVES DE USUARIOS

Añadiendo Usuarios

Slackware Linux hace que añadir un nuevo usuario sea bastante fácil. Todo lo que tiene que hacer es ejecutar el scritp **/usr/sbin/adduser**. Él le presentará una serie de preguntas y luego completara todos los pasos necesarios para poner a funcionar la cuenta. Después de que la cuenta haya sido creada, se pueden usar los siguientes archivos para manejarla:

- **/usr/bin/passwd** - Cambia la contraseña
- **/usr/bin/chfn** - Cambia la información de contacto.
- **/usr/bin/chsh** - Cambia el shell del usuario.

Borrando Usuarios
Para borrar a un usuario tendrá que editar algunos archivos y eliminar algunas cosas, pero es realmente simple. Los siguientes pasos son necesarios para borrar a un usuario de su sistema.

1. **Borre la línea en /etc/passwd**. Como root abra el archivo /etc/passwd y consiga la línea correspondiente al usuario que esta eliminando y bórrela.
2. **Borre el nombre de usuario de /etc/group**. Hay que quitar el nombre de usuario de cualquier grupo donde aparezca en /etc/group.
3. **Borre la línea en /etc/shadow**. El mismo proceso del paso 1.
4. **Elimine el directorio Home.** Como root ejecute el comando rm -rf en el directorio home de la cuenta.
5. **Borre el archivo de mail spool.** Como root se tendrá que borrar el archivo var/spool/mail/{USERNAME}. Otra forma de eliminar los usuarios es utilizando el comando userdel (ubicado en /usr/bin).Se puede usarlo para hacer todo lo de arriba. Ver la página del manual (man) para más información.

Desactivando Usuarios
Desactivar una cuenta es fácil y en algunas ocasiones es preferible.
Las cuentas de los usuarios pueden ser bloqueadas usando las etiquetas -l y -u. La opción -l desactiva una cuenta cambiando el password a un valor encriptado imposible de descifrar. La opción -u reactiva la cuenta devolviendo el password al valor anterior.

1.2.6 COMANDOS BÁSICOS PARA LA INSTALACIÓN DE LINUX SLACKWARE.

- adduser: Sirve para añadir nuevos usuarios al sistema, sólo puede ser ejecutado por el root .
- man: Básicamente la ayuda de Linux, tecleando man y el nombre del comando a buscar te enseña para que sirve ese comando y su forma de utilización.
- ls: Igual que el dir en dos.
- mkdir: crea una carpeta como md en dos.
- rmdir: Borra la carpeta sólo si esta vacía.

- cd: Igual que en dos salvo que si se escribe sin indicar la carpeta a la que se quiere ir, va directo a la carpeta home.
- cp: Sirve para copiar archivos , con el modificador -r copia un directorio completo.
- cat: Equivale al TYPE de dos, sirve para visualizar archivos de texto, sólo visualizarlos.
- pwd: Sirve para indicar en que directorio se encuentra a no ser que el promt este configurado para mostrar la ruta completa.
- setup: Sale un menú para poder configurar las X-Windows, el ratón, las tarjetas de sonido, las particiones, etc...
- startx: Inicia las X-Windows, es decir el entorno gráfico.
- kde: En caso de tener instalado este entorno gráfico lo inicia.
- shutdown: Sirve par cerrar el sistema, debe ser ejecutado con modificadores:
- -g 0: Indica que el sistema se apagara en 0 minutos, el 0 puede ser sustituido por cualquier número entero para retardar el apagado del sistema.
  -r 0: Indica que el sistema se reiniciará en 0 minutos, que se puede cambiar por cualquier número entero como en el anterior modificador.
- reboot: Sirve para reiniciar el sistema inmediatamente.
- alias: Sirve para crear nuevos comandos a partir de uno ya existente (el antiguo sigue teniendo la efectividad de antes), por ejemplo: alias apagar='shutdown -g 0', con lo que apagar y shutdown -g 0 servirían para lo mismo.
- mount: Sirve para montar particiones del disco duro, unidades de disco en general (cd, disquetes, zip, etc..) Ejemplos: mount -t(indica que seguido va el tipo de formato de archivos) iso9660 (formato de archivos) /dev/cdrom (indica la partición que quieres montar) /mnt/cdrom (y el lugar donde se montar) -r -ro (indica que se debe montar como sólo lectura). Como montar:
  Un Diskette: (si tiene formato de dos o windows): mount -t iso9660 /dev/fd0 /mnt/floppy
  Una partición de dos: mount -t fat /dev/hd** (los ** se sustituirán por la posición del dispositivo que se explicó más arriba en el segundo párrafo) /mnt/*** (el directorio que se haya creado para montar esta partición)
- df: Sirve para comprobar el espacio disponible y utilizado en las particiones que están montadas en ese momento.

1.3 EDITORES PARA LA CREACIÓN DE PÁGINAS WEB

Cuando se esta navegando por Internet y se pone en nuestro browser una dirección Web, se están realizando las siguientes tareas:

1. Se envía un requerimiento al servidor Web, el cual consiste en la petición del archivo que se desea ver, por ejemplo la página index.html.
2. El servidor lee el requerimiento, busca la página Web solicitada, y la envía al cliente.
3. El cliente recibe la página en su navegador, que la muestra al usuario.

La figura 3.1 muestra un diagrama a bloques de una arquitectura cliente-servidor.

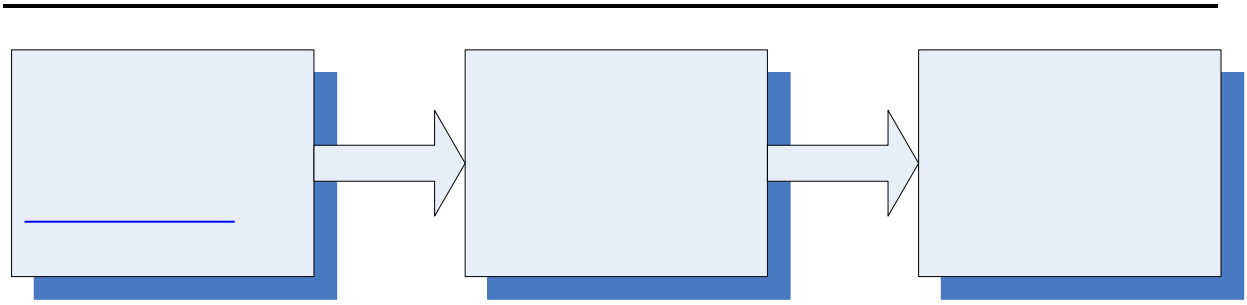


Figura 3.1 Diagrama a Bloques Cliente-Servidor

1.3.1 ETIQUETAS HTML

HTML esta basado en etiquetas: directivas especiales que permiten que el navegador sepa cómo quiere que se muestren los elementos en las páginas. Las etiquetas HTML están encerradas entre símbolos de mayor y menor (<>). Por ejemplo, en el mismo comienzo y final de una página HTML, necesitaría tener las etiquetas <HTML> y </HTML> respectivamente. Por convención, las etiquetas que precisan una etiqueta de apertura y otra de cierre (como sucede con la etiqueta <HTML>) son idénticas excepto por la barra en la etiqueta de cierre. Dentro de las etiquetas <HTML> y </HTML>, todas las páginas deberían tener las etiquetas <HEAD> y </HEAD> y <BODY> </BODY>. Las etiquetas <HEAD> y </HEAD> contienen etiquetas <META>, que ofrecen información sobre lo que mostrara la página (estas etiquetas son útiles para herramientas automáticas que procesan la Web, tales como páginas para incluirlas en los resultados de los motores de búsqueda). Las etiquetas <TITLE> </TITLE> marcan el comienzo y el final de la página. Aparecerá en la barra de titulo de la página.

Con la excepción de las etiquetas <META>, todas las etiquetas descritas hasta ahora se utilizan siempre en pares, con una etiqueta de comienzo y otra de fin. No todas las etiquetas precisan una etiqueta de cierre. La más común es la etiqueta de asalto de línea,
. Con algunas etiquetas, la etiqueta de cierre es opcional. Por ejemplo, hay diseñadores que no utiliza la etiqueta de cierre de las etiquetas de párrafo <P></P>.

HTML no diferencia entre mayúsculas y minúsculas.

1.3.2 CABECERAS

Para dividir correctamente en secciones la página HTML, tal y como se hace en un documento de texto normal, se utilizan las cabeceras, hay de 6 diferentes tamaños: <H1></H1>, <H2></H2>, <H3></H3>, <H4></H4>, <H5></H5> y <H6></H6>, el tamaño se va incrementando del 6 al 1.

1.3.3 PÁRRAFOS

La etiqueta <P> introduce una línea en blanco de párrafo. Esta etiqueta admite su correspondiente cierre </P>, aunque no es necesario.

Para centrar el texto se utiliza <CENTER> aquí va el texto </CENTER>.

1.3.4 LISTAS

HTML ofrece varios mecanismos para especificar listas de información. Todas las listas deben contener uno o más objetos de lista. Las listas pueden contener:

- Listas no numeradas.
- Listas numeradas.
- Listas anidadas.
- Definiciones.

Listas no numeradas.

Comienza el listado con la etiqueta <UL> (Unordered List) y su final con la etiqueta de cierre </UL>. Cada objeto que forma la lista va precedido de la etiqueta <LI> (List Item) sin etiqueta de cierre.

Listas numeradas.

Comienza el listado con la etiqueta <OL> (Ordered List) y su final con la etiqueta </OL>. Cada objeto que forma la lista va precedido, igual que en las anteriores de la etiqueta <LI> sin cierre.

Listas anidadas.

Se pueden combinar unas listas dentro de otras: No numeradas y/o numeradas. Hay que tener, simplemente, cuidado en la colocación de las etiquetas de apertura y cierre.

Listas de definiciones.

Son apropiadas, como su nombre indica, para establecer listados de términos con sus definiciones. Las etiquetas de apertura y cierre son <DL> y </DL> (Definition List). Los conceptos a definir van con la etiqueta <DT> (Definition Term) y las definiciones con la etiqueta <DD> (Definition Definition).

1.3.5 TEXTO PREFORMATEADO

Negrita: Para remarcar una cadena de caracteres empleamos las etiquetas <B> y </B> (Bold).

Cursiva: Para hacer que un texto se muestre en cursiva: <I> e </I> (Italic).

Subrayado: Para subrayar se emplea <U> y </U> (Underlined).

Línea de separación: Se consigue con la etiqueta <HR> (Horizontal Rule) sin etiqueta de cierre.

Imágenes: La etiqueta para introducir una imagen en la página es <IMG src="nombre.gif"> (src de source en inglés fuente u origen de la imagen), y siendo nombre que tiene la imagen y .gif su formato.

La etiqueta funcionará bien siempre y cuando la imagen esté en la misma carpeta en la que se encuentra la página Web desde la que se llama. Es habitual (y recomendable) colocar todas las imágenes en otra carpeta interna, en ese caso, habrá que darle la ruta en la cual se encuentra la imagen. Suponiendo que la imagen nombre.gif se encuentra en una carpeta denominada img, pues bien, la etiqueta sería <IMG src="img/nombre.gif">.

1.3.6 VÍNCULOS HTML

HTML permite crear vínculos, también conocido como hipervínculos, dentro de las páginas. Los vínculos se marcan con etiquetas <A></A>. Por ejemplo, para crear un vínculo a una página llamada test.php, se podría utilizar un código como el siguiente:

Para acceder a test.php, <A HREF=”test.php “>Pulse aquí</A>.

El vínculo ***Pulse aquí*** aparecería en la mayoría de navegadores como una línea subrayada, aunque se puede controlar la apariencia que mostrarán los vínculos. En general, el cambio en la apariencia de un vínculo no es una buena idea. Los usuarios han crecido esperando que palabras y frases subrayadas les permitan saltar a otra página, por lo que trabajar contra éstas expectativas puede conducir a la frustración y confusión. La etiqueta <A> se puede utilizar como un vínculo o un delimitador que define la sección con nombre de un documento. Si la etiqueta <A> tiene un atributo *NAME* pero no uno *HREF,* es un delimitador[2]. Si están presentes ambos campos, la etiqueta es tanto un delimitador como un vínculo.

Si quisiera saltar a otra página pero también pasar un argumento que se pudiera utilizar en la página de destino, podría utilizar el siguiente código:

Para acceder a test.php, <A HREF=”test.php?arg=1”>Pulse aquí</A>.

Para pasar múltiples parámetros, se utiliza la misma sintaxis, pero con un carácter & entre los

[2] Un delimitador es un destino dentro de una pagina Web al que se puede acceder desde un vinculo. Podemos poner un delimitador en cada uno de los párrafos de un sitio Web y utilizar los vínculos para acceder a estos delimitadores que indican los párrafos

parámetros, de la siguiente forma:
Para acceder a test.php
<A HREF=”test.php?arg=1&arg2=2>Pulse aquí</A>.

1.3.7 FORMAS, GET Y POST.

HTML proporciona cuadros de texto, listas desplegables, cuadros de lista, casillas de verificación y botones de opción

Todos los objetos gráficos están contenidos entre las etiquetas <FORM></FORM>. Los elementos gráficos de HTML fuera de un formulario no tienen valor alguno y generalmente no harán lo que quieren que haga. Una etiqueta <FORM> también puede contener un atributo que describe qué <<acción>> (atributo action) tomar cuando se envía el formulario así como un atributo method (<<método>>) para especificar cómo se va transmitir la información del formulario del servidor. El atributo method puede tomar dos valores *Post* o *Get*. El valor predeterminado es *Get*; sin embargo *Get* está descartado en HTML 4 por problemas de internacionalización. El método Post es un método de dos pasos (transparentes para el desarrollador de la página) que envía toda la información que se introduce en el formulario a una localización estándar donde el servidor la lee. El método *Get* anexa el contenido del formulario a la URL como argumentos. Por ejemplo, si un formulario está utilizando el método *Get* y la acción de una página llamada test.php, ésta es la URL resultante que aparecerá en la ventana de la localización (exceptuando cualquier error de procesamiento):

http://<host>/<directory>/test.php?name=Luis

¿Cuál es el mejor método a utilizar, Post o Get?

No hay respuesta sencilla. Para pequeños formularios que envían y reciben pocos datos, *Get* puede ser más eficiente. Para formularios mayores, algunos servidores fallarán si la URL es demasiado larga y, por lo tanto, es mejor utilizar el método *Post*. Además si se va a enviar una clave de vuelta al servidor, debería utilizarse *Post* para que la clave no aparezca en la URL como texto simple.

1.3.8 TABLAS HTML

Para conseguir los alineamientos, en las páginas de HTML, se utilizan las tablas. La etiqueta <TABLE> marca el comienzo de una tabla HTML, en ocasiones no es obvio darse cuenta del uso de ellas, esto es porque hay muchos atributos que pueden controlar como se muestran las tablas. Cada fila de la tabla está englobada dentro de etiquetas <TR></TR>. Estas filas de tablas tienen atributos adicionales para controlar el color de fondo, alineación y otras propiedades. Dentro de cada fila, las etiquetas <TD></TD> controlan las columnas individuales de la fila. Para alinear los elementos gráficos se puede utilizar el valor del atributo width (<<ancho>>) de la etiqueta <TD>.

Un problema de las tablas HTML es que, si se omite la etiqueta de cierre o se anidan las etiquetas de forma incorrecta, los resultados variarán dependiendo del navegador que se utilice.

1.4 ADMINISTRACIÓN DE SERVIDORES WWW

1.4.1 DEFINICIÓN DE UN SERVIDOR WWW

Un servidor web es un programa encargado de ofrecer comunicación mediante el protocolo HTTP (hypertext transfer protocol).[3]

Un servidor web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP que se suele conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita como se puede ver en la figura 4.1.

Figura 4.1 Diagrama a bloques del funcionamiento de un servidor www

El cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Las aplicaciones de servidor suelen ser la opción por la que se opta en la mayoría de las ocasiones para realizar aplicaciones web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional, como sí ocurre en el caso de querer ejecutar aplicaciones javascript o java. Así pues, cualquier cliente dotado de un navegador web básico puede utilizar este tipo de aplicaciones.

1.4.2 FUNCIONAMIENTO

Cliente Web HTTP Int

El protocolo http, como su nombre le indica, fue diseñado para transferir documentos de hipertexto (documentos HTML). En sus orígenes cuando Internet era ARPANET y era muchísimo más pequeña de lo que es ahora, y cuando la velocidad de conexión era enormemente inferior, texto plano era lo único que se transfería por ella. A nadie se le ocurria siquiera publicar un documento con imágenes o sonidos. A medida que avanzó se hicieron evidentes y el protocolo fue mejorando para poder transferir cualquier tipo de datos.

[3] Broten Rich Coar Ken, Servidor Apache, (2000), p. 57

HTTP tiene otras varias características pero la que en este caso va a interesar más es ue es un protocolo que no guarda el estado (stateless). Esto significa que no se mantienen una conexión constantemente con el cliente y el servidor, sino que el cliente manda el pedido y corta la conexión, sin guardar información sobre los pedidos anteriores. De esta forma, el servidor trata cda pedido en forma independiente del anterior, simplemente porque no puede saber si el pedido proviene del mismo cliente, aunque hayan ocurrido muy cerca en el tiempo.

Se puede resumir en 4 pasos el funcionamiento del protocolo HTTP:

1.- El cliente HTTP abre una conexión.
2.- El servidor manda una "acknowledge".[4]
3.- El cliente envía su request message.
4.- El servidor responde "response message".

Lista de los servidores web más populares:

- Apache
- BadBlue
- IIS (Microsoft).
- Sun Java Web Server
- Sambar Server
- SimpleServer:WWW
- Xitami
- Zeus

1.4.3 ¿POR QUÉ APACHE?

Apache es rápido, fiable y barato. Apache puede ser todo esto porque es open source. Esto significa que tiene acceso al código fuente.

De acuerdo a las estadísticas de Netcraft2 (http://netcraft.com/) el servidor Web Apache se emplea más que el resto del conjunto de servidores Web.

El servidor Web Apache es un servidor HTTP (Web) gratuito totalmente equipado, desarrollado y mantenido por el Apache Server Project (proyecto servidor Apache). El objetivo del proyecto es ofrecer un servidor Web fiable, eficiente, y fácilmente extensible, con código fuente abierto gratuito. El software del servidor incluye el demonio de servidor, archivos de configuración, herramientas de administración y documentación. El Apache Server Project es uno de los proyectos actualmente dirigidos por la Apache Software Foundation. Esta organización sin ánimo de lucro ofrece soporte legal, financiero y organizativo para varios proyectos de código abierto de Apache, entre los que se encuentran Apache HTTPD Server, Java Apache, Yakarta y XML-Apache.

[4] Un acknowledge es un mensaje para verificar la transferencia de un bloque datos satisfactoriamente.

Tabla 4.1 Servidores web más populares.

Top Developers					
Developer	***October 2005***	***Percent***	***November 2005***	***Percent***	***Change***
Apache	52005811	69.89	52928740	70.98	1.09
Microsoft	15293030	20.55	15096547	20.24	-0.31
Sun	1889989	2.54	1879576	2.52	-0.02
Zeus	585972	0.79	579776	0.78	-0.01

1.4.4 COMPILAR E INSTALAR APACHE

Apache está disponible en forma binaria en varias plataformas, pero normalmente está disponible como código fuente. Esto significa que tendrá que tener un compilador C y compilar e instalar Apache.

Requerimientos del sistema

Como mínimos 12 MB de espacio temporal en la unidad de disco duro para el proceso de instalación. Tras la instalación, Apache ocupa cerca de 3MB, además del espacio que se utilice para colocar el contenido Web.

También un compilador ANSI-C. El compilador GNU C, que se conoce como GCC, es el compilador recomendado, pero otros compiladores también trabajan bien si son compatibles con ANSI-C.

Conseguir Apache

El software de Servidor Apache está disponible en el sitio web del Grupo Apache y en decenas de sitios mirrors de todo el mundo.
Los URL importantes son:

- La fundación Apache Software, que está en http://www.apache.org/. Hay más de un proyecto bajo la protección de la ASF, aunque Servidor Apache es el más conocido.
- El proyecto Apache http Server, en http://www.apache.org/httpd.html. Este sitio es la fuente de información más exacta y actualizada que existe sobre Servidor Apache. Toda la documentación del servidor está disponible en línea, así como la base de datos de fallos, los archivos de noticia, la información histórica y otros tipos de recursos relacionados con Apache.
- Descargar apache en http://www.apache.org/dist. Esta es la ubicación principal para obtener el código fuente de Apache.
- Los mirrors del proyecto Apache en http://www.apache.or/mirrors/. Este sitio enumera, por código del país, los sitios mirrror oficiales de Apache.

Apache está disponible para su descarga en varias versiones, en código binario y en código fuente, en la página de descarga de Apache y en los distintos mirrors.

Descargar la última versión siempre es lo más seguro, ya que el Grupo Apache prueba el software antes de que se pueda descargar. Sin embargo se debe de leer la lista de fallos conocidos, para así poder estar al tanto de los temas problemáticos con el software y para evitar una versión que pueda tener un problema que la afecte directamente.

Descargar Binarios.

Apache está disponible en forma binaria en una serie de plataforma, antes de usar un binario, hay que asegurarse de que ha sido construido con las opciones que interese utilizar. Si se desea una construcción genérica sin módulos opcionales, probablemente convenga utilizar uno de éstos. Asegúrese de que lo construye con una configuración que coincida con la suya, para no encontrar así problemas de compatibilidad. Siempre es mejor construir Apache a partir del código fuente.

Descargar código fuente

Si se va a construir Apache desde el código fuente, se recomienda descargar el archivo .tar.gz correspondiente a la versión que se haya decidido utilizar, por ejemplo en este caso se descargo httpd-2.0.54.tar.gz

Verificar la autenticidad del archivo

Es importante verificar que el archivo descargado sea un articulo genuino, si es que se descarga el archivo desde un sitio mirror en vez del sitio Web Apache..

Se podrá verificar el archivo con diferentes programas por nombrar alguno pretty Good Privacy.

Extraer el contenido del archivo

Para desempaquetar este archivo, se utiliza el siguiente comando:

tar –zxvf httpd-2.0.54.tar.gz

Después de desempaquetar, se va a crear un archivo llamado httpd-2.0.54 (va a depender de la versión que este utilizando).

Instalación para usuarios impacientes

Para resumir la instalación de apache, se tienen que seguir los siguientes pasos: se tiene que estar conectados como *root* para ejecutar éstos comandos eficientemente:

```
cd httpd-2.0.54
./configure –prefix=/usr/local/apache
make
make install
/usr/local/apache/apchectl star
```

Se puede cambiar el prefijo a otra cosa si es que se desea instalarlo en otro lugar que no sea /usr/local/apache. Ésta es la ubicación predeterminada para instalar Apache. Make se utiliza para compilar Apache.

1.4.5 INICIAR, DETENER Y REINICIAR EL SERVIDOR

Dependiendo del sistema operativo que se este ejecutando, se puede iniciar, detener y reiniciar el servidor Apache de varias maneras. En Linux, el script del shell **apachectl** le da la posibilidad de iniciar, detener y reiniciar Apache en la línea de comandos.

Iniciar el servidor

En Linux puede iniciar Apache en la línea de comandos escribiendo el nombre del ejecutable **httpd**, con las opciones de línea de comando que desee. También puede dejar que Apache se abra automáticamente al iniciar el sistema, el *script* **apachectl** puede resultar muy útil a la hora de proporcionar ésta funcionalidad, ya que acepta **star** y **stop** como argumentos, lo que se espera en los *scripts* **/etc/rc.d** de los tipos de Linux que soporten estos mecanismos.

Detener Apache

Para detener Apache, emita el comando Hill –TERM en el ID de proceso que se enumeran en el archivo httpd.pid

Reiniciar Apache

Existen dos formas de reiniciar el servidor Apache, dependiendo de la rapidez con la que se quiera ejecutar el reinicio.

- Para reiniciar inmediatamente, utilice una señal HUP
- Para reiniciar suavemente, utilice la señal USR1

El script apachectl

En una instalación Linux de Apache, hay un script de shell llamado **apachectl**, que le evitará tener que recordar la multitud de formas de iniciar, reiniciar y detener el servidor, apachectl debe incluirse en el directorio **/src/support** de la distribución Apache; una vez que se construye Apache, éste contendrá las rutas correctas de todo lo que contiene su sistema.

El uso de **apachectl** es muy claro. La simple acción de ejecutar **apachectl** en la línea de comandos (o con el argumento help) le presenta todas las opciones disponibles. Apachectl es, sencillamente, un script **/bin/sh** que contiene algunas opciones de línea de comandos y funciones kill.

Configurar Apache

Apache proporciona al administrador del servidor, mucho control sobre las distintas formas en que va a funcionar el servidor, permitiéndole controlar todo, desde el aspecto de los mensajes de error hasta la cantidad total de tiempo de la CPU que puede consumir el servidor. En la mayoría de casos, los valores predeterminados serán los adecuados.

El comportamiento del servidor Apache se define en el archivo de configuración del servidor httpd.conf.

Tradicionalmente, la configuración de Apache ha estado dividida en tres archivos de configuración: httpd.conf, access.conf y srm.conf. Con el tiempo, la distinción de lo que entraba en un archivo o en otro se fue haciendo más borrosa y, así a partir de la versión 1.3.4 de Apache, los tres archivos están combinados en un sólo archivo de configuración.

Con la antigua forma de proceder, httpd.conf era el archivo de configuración del servidor principal, access.conf era el archivo que definía los permisos de acceso y srm.conf definía los recursos del servidor, como las asignaciones de directorios y los iconos. La documentación de los antiguos servidores hace referencia a estos archivos, por lo que es útil saber para qué servían.

1.4.6 DIRECTIVAS DE CONFIGURACIÓN

Las directrices están ubicadas en los archivos de configuración, a partir de la versión 1.3.4, Apache recomienda que todas las directrices de configuración estén en un sólo archivo, el archivo httpd.conf. Con las directrices, puede introducir información de configuración básica, como el nombre del servidor, o llevar a cabo operaciones más complejas, como implementar host virtuales. Apache tiene una gran variedad de directrices diferentes que ejecutan operaciones tan diversas como controlar el acceso a directorios, asignar formatos de iconos de archivos o crear archivos de registro. A continuación se describen las directivas de configuración disponibles (Tabla 4.2).

Tabla 4.2 Directrices de configuración de Apache

Directriz	*Descripción*
AccessConfig *archivo*	Archivo que deberá leerse en busca de más directrices después de que se lea el archivo ResourceConfig *Archivo* es el relativo a ServerRoot. Valor predeterminado: AccessConfig **conf/access.conf** Contexto: Configuración del servidor, hosts virtuales.
Action *tipo_de_accion script-cgi*	Añade una acción, que activa el *script-cgi* cuando *tipo_de_accion* sea activada por la petición. Contexto: configuración del servidor, hosts virtuales, directory, **.htaccess.** Modifica: FileInfo. Módulo: mod_actions.
AddDescription *cadena archivo archivo..*	Establece la descripción que se mostrará para un archivo. Puede ser una extensión, un nombre de archivo parcial, una expresión con comodines, o un nombre completo. La cadena debe estar entre comillas dobles (“). Contexto: configuración del servidor, hosts virtuales, directory, **.htaccess.** Modifica: Indexes. Estado: Base. Módulo: mod_autoindex.
AddModule *módulo módulo*	Habilita el uso de módulos compilados, pero no en uso. Contexto: configuración del servidor.

Directriz	*Descripción*
Anonymous *usuario usuario…*	Usuario a los que se autoriza el acceso sin verificación de contraseña. El *usuario* es, normalmente, anonymous (distingue mayúsculas de minúsculas). Valor predeterminado: none. Contexto: directory, **htaccess.** Estado: extensión. Módulo: mod_auth_anon.
AddType *tipo-MIME extensión extensión..*	Asigna las extensiones dadas al tipo de contenido especificado. Se usara *tipo-MIME* para los archivos con dichas extensiones. Contexto: configuración del servidor, hosts virtuales, directory, **.htaccess** Modifica: FileInfo. Estado: Base Módulo: mod_mime
AuthName *dominio-auth*	El ámbito de autorización para un directorio. Se le da un ámbito al cliente para que el usuario sepa qué nombre de usuario y contraseña debe enviar. Contexto. Directory, **.htacces.** Modifica: AuthConfig
AuthUserFile *archivo*	Asigna el nombre del archivo dado a la lista de autenticación de usuarios y contraseñas. Contexto: directory, **.htaccess.** Modifica: AuthConfig. Estado: Base. Módulo: mod auth.
<Directory *directorio***>** … **</Directory>**	Las directrices **<Directory> y </Directory>** operan como etiquetas que encierran un grupo de directrices aplicables solamente al directorio especificado y sus subdirectorios. Contexto: configuración del servidor, hosts virtuales.
DocumentRoot *nombre-de-directorio*	El directorio desde el que **httpd** sirve archivos. Valor predeterminado: DocumentRoot **/usr/local/apache/ htdocs (/var/www/html** en sistemas Red Hat y Caldera). Contexto: configuración del servidor, hosts virtuales.
ErrorDocument *código-error documento*	Redirige a una URL local o externa para tratar el error o problema. Contexto: configuración del servidor, hosts virtuales, directory, **.htaccess.**
Group *grupo-unix*	Establece el grupo para el servidor. Un servidor autónomo deberá ejecutarse inicialmente como root. Se recomienda que configure un grupo nuevo específicamente para ejecutar el servidor. Valor predeterminado: Group #-1. Contexto: configuración del servidor, hosts virtuales.

Directriz	*Descripción*
IndexOptions *[+\-]opción [+\-]opción*	Establece opciones para indexación de directorios: Fancylndexing, IconHeight, IconsAreLinks, IconWidth, NameWidth, ScanHTMLTitles, SuppressDescription. Contexto: configuración del servidor, hosts virtuales, directory, .htaccess. Modifica: Indexes. Estado: Base. Módulo: mod_autoindex.
Listen *[dirección IP:] número de puerto*	Escucha en más de una dirección IP o puerto. Por omisión, cursa peticiones en todas las interfaces IP, pero solamente en el puerto especificado en la directriz Port. Contexto: configuración del servidor.
LoadModule *módulo archivo*	Crea un enlace para el archivo o biblioteca especificados y añade la estructura modular *módulo* a la lista de módulos activos. Contexto: configuración del servidor. Estado: Base. Módulo: mod_so.
Options *[+\-]opción [+\-]opción ...*	Controla las características del servidor disponibles en un directorio en particular. Si se establece None (ninguna), no se habilitará ninguna característica extra. All Todas las opciones, excepto en el caso de MultiViews. Ésta es la configuración predeterminada. ExecCGI Se permite la ejecución de scripts CGI. **FollowSymLinks** El servidor sigue enlaces simbólicos en este directorio. **Includes** Se permiten inclusiones desde el servidor. **IncludesNOEXEC** Se permiten inclusiones desde el servidor, pero se inhabilitan los comandos #exec y el #incluye de los scripts CGI. **Indexes** Devuelve una lista formateada del directorio, en directorios sin Directorylndex. MultiViews Se permiten MultiViews de contenido negociado. **SymLinksIfOwnerMatch** El servidor sólo seguirá enlaces simbólicos para los que el archivo o directorio destino tengan el mismo propietario que el identificador de usuario del enlace. Contexto: configuración del servidor, hosts virtuales, di-rectory, **.htaccess**. Modifica: Options.
order *ordenación*	Controla el orden por el que se evalúan las directrices allow y deny. Valor predeterminado: order deny, allow. Contexto: directory, .htaccess. Modifica: Limit. Estado: Base. Módulo: mod_access.

1.5 PROGRAMACIÓN CON PHP

PHP también denominado PHP Hypertext Preprocessor, es un lenguaje de programación interpretado de alto nivel para Internet, muy similar en su sintaxis al lenguaje C, Java o Perl, con algunas diferencias, no compila como el lenguaje C, es un intérprete, por lo tanto cada vez que debe ejecutarse un programa, lo interpreta, verificando toda su sintaxis.

El objetivo del lenguaje PHP es brindarles a los creadores de sitios Web la posibilidad de desarrollar sitios dinámicos en forma sencilla y rápida, aunque las posibilidades y funcionalidades de PHP son muy superiores al simple hecho de sólo hacer una página Web dinámica.

En Internet ya no basta con tener un sitio estático porque se esta limitado para introducir cambios en su contenido en tiempo real, es decir, que cada vez que se requiere introducir una modificación, se debe crear la página Web para hacerlo, y luego publicarla al servidor donde reside nuestro sitio Web. En un sitio dinámico, esta operatoria cambia radicalmente, ya que la información del sitio, generalmente está contenida en una base de datos. Cada vez que se muestra una página, como una de noticias, se busca en la base de datos las últimas noticias que se tienen ingresadas para mostrar en el navegador del visitante. Ahora bien, si se quiere que la página muestre nuevas noticias, simplemente se cargan las noticias en la base de datos, por ejemplo, a través de un formulario, siendo esto el único cambio que se debe hacer, ya que desde el momento en que esté la noticia cargada, automáticamente aparecerá en la página de noticias. Imaginemos por un instante la operatoria de un periódico on-line:

De manera estática, todos los periodistas escriben una nota en un procesador de palabras, como por ejemplo Word, que después envían al editor, para que le de su aprobación. Luego, una vez que está aprobada, le reenvían la nota al webmaster o diseñador del sitio, quien se encargará de diseñar una nueva página Web con dicha nota, y así subirla al servidor como un nuevo contenido del sitio Web, además de vincularla, por ejemplo a la página principal del periódico.

En un sitio dinámico, todos los periodistas, acceden en forma remota, desde cualquier parte del planeta donde dispongan de un equipo con acceso a Internet, a un menú, mediante usuario y una contraseña, desde donde envían la nota que han escrito. Ésta es grabada es una base de datos junto a los datos del autor en un estado, por ejemplo, "Pendiente de Publicación", y automáticamente terminado este proceso se le envía un correo electrónico al editor para que se conecte a la interfaz de administración del periódico, mediante un usuario y contraseña para poder revisar y aprobar la noticia escrita por el periodista. Cuando el editor aprueba la nota, automáticamente estará disponible en el sitio Web.

En este ejemplo del periódico on-line, las ventajas de un sitio dinámico son altamente superiores, ya que permiten, además de automatizar tareas, poder generar un buen feedback con los visitantes, mostrar un sitio atractivo de actualización permanente, e inclusive automática.

PHP es una excelente herramienta para poder realizar sitios dinámicos, que deben de conectar con Base de Datos, y que ha recibido una gran aceptación por parte de la comunidad de desarrolladores de Internet, ya que se esta haciendo muy popular. En la actualidad hay más de tres millones de sitios Web desarrollados con esa tecnología, que va ganando más adeptos día a día. Es sumamente funcional, es gratuito y open source, por lo que está testeado permanentemente por

miles de desarrolladores en el mundo, que le van aportando diversas funcionalidades día a día.

PHP es gratuito, y es un programa de código abierto (open source), el cuál está disponible para cualquier persona. Esta es la ventaja principal, por la cual PHP se ha hecho uno de los lenguajes más usados del mundo. Bastará que se ingrese al sitio Web de PHP a la sección de Downloads para bajarlo y listo. Acerca de la licencia de PHP, podrán consultarla desde el sitio oficial de PHP: www.php.net

1.5.1 LENGUAJES DE CLIENTE O LENGUAJES DE SERVIDOR

Cuando se arma una página Web, la mayoría de las veces se utilizan los lenguajes de cliente, es decir que cada vez que se va a mostrar la página Web, el archivo de ella se recupera del servidor, pero el programa que se encarga de interpretar dicha página es el cliente, en este caso el navegador Web. Algunos ejemplos de estos lenguajes de cliente son el propio HTML o Javascript, entre otros. En los lenguajes de servidor, las páginas son interpretadas en el propio servidor, tal es el caso de PHP, o de tecnologías similares como ASP, ColdFusion, Java Server Page o Perl, entre otros.

1.5.2 ¿CÓMO FUNCIONAN LAS PÁGINAS PHP?

1 Se envia un pedido al servidor Web para ver una página, por ejemplo index.php.
2 El servidor Web recoge el pedido del cliente y busca la página.
3 Si la página es PHP, llama al intérprete de PHP para ejecutarla, depende de si conecta con una base de datos, en cuyo caso también conectará a la página.
4 Se ejecuta el código en el servidor y se preparan los resultados en formato html.
5 Se envían los resultados al cliente, para que sean interpretados por el navegador.

La figura 5.1 muestra en un diagrama a bloques los pasos anteriores.

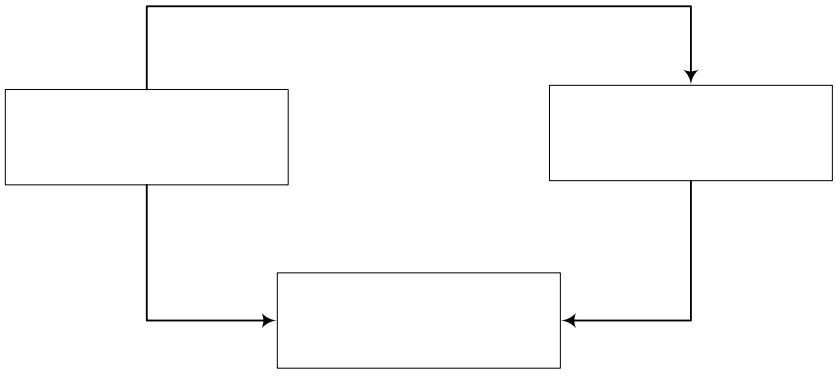


Figura 5.1 Diagrama a bloques del funcionamiento de las páginas PHP

1.5.3 ¿QUÉ SE NECESITA PARA PODER PROGRAMAR CON PHP?

Para poder utilizar PHP, se necesitará en principio un Servidor Web, quien va a manejar las peticiones de los usuarios, y que puede ser el Servidor Web Apache que es uno de los más utilizados a nivel mundial y viene disponible en versiones para Windows y para Linux.

También se necesita tener instalado PHP, y en el caso de que se trabaje con alguna base de datos, por ejemplo MySQL, también se debe instalarla.

Al momento de comenzar a programar, se debe tener levantado (en funcionamiento) el servidor Web Apache, y si se trabaja con MySQL, también deberá estar corriendo el proceso **mysql.**

1.5.4 ¿QUÉ SE PUEDE HACER CON PHP?

Lo más importante que se puede hacer con PHP será pasar de sitios Web estáticos a sitios Web dinámicos, y conectarlos con base de datos, como MySQL, Oracle, Sybase o Microsoft SQL Server entre otras. También se podrán procesar formularios, hacer otra cosa, como guardarla en una base de datos, o enviarla por e-mail. A su vez, PHP también permite, si se tiene instalada la librería de imágenes GD, generar imágenes en tiempo real, poder modificarlas, agregarle texto, reducirlas, etc. También se dispone de muchas otras funciones para el manejo de archivos y directorios, ya sea para crear nuevos archivos o para acceder al contenido de ellos y poder luego modificarlos y grabarlos.

1.5.5 CONCEPTOS BÁSICOS

Sintaxis

De la misma forma que los documentos HTML, están estructurados por TAGS, lo mismo sucede cuando se escribe en PHP, ya que, en sí un programa de PHP, que se verá a través del browser de Internet, es código HTML + Código PHP. Indicarle al documento que se va a comenzar a escribir en código PHP es similar a como se hace cuando se inserta un subprograma Java o similar. Se debe incluir los tags de apertura de código y los tags de cierre.

Los Tags son los siguientes:

Inicio de Código PHP: <?
Fin de Código PHP: ¿>

También se deberá respetar siempre las siguientes pautas:

- **Todas las operaciones deben terminar con : ;** (punto y coma).
- **Comentarios:** Se definen anteponiendo una doble barra o encerrando el comentario entre /* y */, por ejemplo:

  /*Esto es un comentario*/

Esta última forma es muy útil cuando debemos comentar varias líneas.
Veamos sobre una página Web cuáles son las diferencias.

El código PHP se puede introducir en cualquier porción de la página Web, ya sea en el título, el cuerpo, etc.

1.5.6 VARIABLES Y TIPOS DE DATOS

Las principales variables de éste lenguaje PHP soporta los siguientes tipos de variables:

Integer: por ejemplo1.
Double: por ejemplo 1.3
String: por ejemplo "Mi primer página PHP".
Array: por ejemplo guardar valores en una misma variable.
Object: para guardar objetos.

Creación e inicialización de variables

Todas las variables inician con el signo $, por ejemplo:

```
$contador = 123;
```

La variable contador vale 123 y es un integer.

```
$contador = "123";
```

La variable contador vale 123 y es un texto, esto es así porque lo hemos puesto entre comillas.

```
$contador = 1.23;
```

La variable contador en este caso vale 1.23 y es double.

A diferencia de otros lenguajes, las variables no necesitan ser declaradas en forma precisa. Simplemente será necesaria la asignación de un valor a la variable y listo. Cuando le asignamos un valor a una variable, PHP automáticamente determina el tipo de la variable que es.

Las variables en PHP son del tipo case sensitive, esto quiere decir que distinguen entre mayúsculas y minúsculas.

1.5.7 EXPRESIONES

Al programar se utiliza expresiones en forma permanente. Un ejemplo de expresión sería la definición de una variable.

Al definir una variable, por ejemplo:

```
$nombre = "Cesar"
```

Esta asignando el valor "Cesar" a la variable llamada $nombre.

Las expresiones de las que se disponen son las siguientes:

> mayor a
>= mayor o igual a
== igual a
< menor a
<= menor o igual a
!= distinto

Operadores matemáticos

Los principales operadores matemáticos son los siguientes:

+ Suma
- Resta
* Multiplicación
/ División
% Módulo (calcula el resto de una división)

Operadores de asignación

El operador para asignar es el igual (=), significa que asignamos al operando de la izquierda el valor del operando de la derecha. Hay que tener cuidado de no confundir una operación de asignación con una operación de comparación.

Operadores para concatenar

El operador para concatenar es el . (punto).

Por ejemplo, se tienen dos variables, nombre y apellido, y se requiere juntarlas, se procedería de la siguiente forma (listado 5.3):

```
<?
      //Declaro e inicializo variables

      $nombre = "Cesar";
      $apellido = "Desales";

      Echo($nombre." ". $apellido);
?>
```

Listado 5.3 Ejemplo para concatenar una cadena.

Esto producirá la siguiente salida.

Cesar Desales

Operadores de Comparación

Estos operadores permiten comparar, dos valores, y son muy utilizados, por ejemplo en las condiciones if.

Los más utilizados son:

== igualdad

!= distinto
< menor
> mayor
<= menor o igual
>= mayor o igual

Operadores lógicos

Los operadores lógicos son aquellos que se utilizan para combinar condiciones. Las condiciones con **and** serán verdaderas si se cumplen ambas condiciones. Las condiciones con operadores **or** serán verdaderas cuando al menos una de ellas se cumpla. Los operadores **xor** serán verdaderos si sólo una de las dos condiciones es verdadera, y por último el operador **not**, cambia el valor de la variable.

1.5.8 SENTENCIAS

La programación estructurada nos permite crear condiciones de ejecución para que los programas que desarrollamos hagan diferentes cosas ante diferentes condiciones. Al permitirnos ejecutar sentencias, podremos utilizar la verdadera funcionalidad a nuestra aplicación, ya que esto definirá la recepción de parámetros.

Sentencias Condicionales

Las sentencias condicionales son aquellas que ante el establecimiento de una condición, producirán distintas alternativas de ejecución, dependiendo si la condición se evalúa por verdadero o falso.

If

El **If** es sin duda uno de los componentes más importantes de cualquier lenguaje de programación, ya que si carece de él no se podría realizar comparaciones.
El if pregunta si se cumple una condición, y provee dos salidas, cuando la condición se cumple y cuando la condición no se cumple, utilizando un **else** (sino), o sea, una por verdadero y otra por falso.

Elseif

También se dispone de otra sentencia condicional llamada **elseif,** que permitirá realizar una nueva comprobación en caso de que el resultado de la condición primaria haya sido falso.

Otra forma de escribir una sentencia **if** es utilizando **:** (dos puntos) sin llaves y como finalización del código que se requiere ejecutar, escribir **endif.**

Switch (case)

En muchas ocasiones es necesario que se compare la misma variable contra varias opciones posibles. Esto se podría hacer con un if como se vio anteriormente o también se puede utilizar la sentencia switch.

La sentencia **switch** compara el valor del número contra todas las posibilidades existentes, cuando encuentra que se cumple la condición ejecuta todo el código de esa condición, desde los **:** (dos puntos) hasta que encuentre un **break**; que finaliza la ejecución del **switch**. También se puede

utilizar la opción **default** al final del **switch.**

Bucles de Control o Loops

Un bucle de control o loop sirve en programación para ejecutar n cantidad de veces una porción de código de un programa. Puede ser para ejecutar n cantidad de veces o hasta que se cumpla una condición especifica.

While

Esta instrucción permite ejecutar el código del programa mientras (*while* en inglés) se cumpla la condición. Se evalúa que se cumpla la condición:

- Si la condición es válida ingresa en el loop (el ciclo while) y ejecuta la porción de código que se haya puesto en el mismo.
- Cuando deja de cumplirse la condición, el programa sale del **while** y continúa con el resto del programa.
- Si por el contrario, la condición no se cumple, el código **while**, no se ejecutará.

Esta función es particularmente importante, junto con do..while, o similares, ya que permitirá leer los registros de una tabla cuando se este trabajando con base de datos. De esta forma se podrá listar, por ejemplo todos los nombres que tenga en la base de datos de la agenda.

```
while (condición)
{
        ……
        ……
}
```

```
Otra forma de escribir el while
while (condición):
        ……
        ……
endwhile;
```

Hay que tener en cuenta que en un while debe haber una variable que pueda alterar la condición del mismo, ya que sino se entraría en un loop infinito que no tendría fin.

Do .. while

Esta sentencia es igual a la anterior (while) con la diferencia de que en vez de evaluar si se cumple la condición al principio del bucle, lo evalúa al final de este.

De esta forma, nos aseguramos que todo el código contenido en el do..while, será evaluado al menos una vez, se cumpla la condición o no. La sintaxis es:

```
Do
{
        …….
        …….
}while (condición);
```

For

La sentencia **for** realiza en bucle un determinado número de veces. Encontraremos que se divide en tres partes:

1 En la primera inicializaremos la variable que utilizaremos en la condición.
2 En la segunda es donde debemos indicar la condición por la cuál finaliza el bucle.
3 En la tercera es donde modificaremos la variable, ya que sin ésta volveríamos a entrar a un loop infinito como vimos anteriormente en el **while**.

Estas tres partes van separadas por ; (punto y coma)

También de la misma manera, podemos utilizar la sintaxis alternativa con dos puntos y un **endfor** en reemplazo de las llaves.

Break

La función **break** se utiliza para cortar la ejecución de un código dentro de un bucle.

Continue

Otra sentencia disponible es **continue,** que sirve para saltear en un bucle el resto de interacción y continuar la ejecución al comienzo de otra interacción. De forma similar al **break**, también podemos indicarle cuantos bucles hay que saltearse.

1.5.9 FUNCIONES

Las funciones son partes de código definidas, que pueden ser invocadas desde otra parte de un programa. La ventaja principal que tiene el uso de funciones es que permite realizar un programa en forma modular y no tener que repetir el código varias veces en distintos programas.

¿Para qué sirven?

La razón de ser de las funciones es evitar repetir código, ya que si se tiene que poner el mismo código en varios programas, es más sencillo declarar una función y luego que los programas que la necesiten la invoquen.

Otra ventaja es que si se necesita cambiar algo sobre el funcionamiento de la función no es necesario corregir todos los programas que la utilizan, sino simplemente corregir la función donde está declarada y listo.

A ellas se les pasan una serie de argumentos, los cuales van entre paréntesis. Si la función no lleva argumentos, simplemente se escribe de la siguiente forma: function nombre(). Por el contrario, si la función lleva argumentos, el llamado a esta será.

```
function nombre(argumento1, argumento2, argumento3)
```

Las funciones permitirán escribir menos, trabajar más rápido y poder leer mejor los programas, de una forma más ordenada, ya que llevará menos código.

¿Cómo se declaran?

Las funciones se declaran en PHP utilizando la sentencia **function.** La sintaxis para declarar

una función es entonces la siguiente:

```
function nombre_de_la_función (parámetros_de_la_función)
   {
        Código de la función
   }
```

1.5.10 PASANDO ARGUMENTOS

Para pasarle argumentos a una función, se debe hacer mediante una lista de argumentos, que se pone entre paréntesis, y donde se ingresan todas las variables, separadas por coma. Por ejemplo, llamando a la función mayúsculas.

```
mayusculas("Juan","Pérez");
```

La información que se envía a las funciones puede ser mediante una lista de parámetros, una lista de variables o constantes separadas por coma.

Por defecto, todos los argumentos pasados a las funciones son por valor, ésto significa que si se cambia el valor del argumento dentro de las funciones, no se verá modificado fuera de dicha función. Si en cambio se quiere cambiar los valores, se debe pasar por *referencia*, anteponiendo un & (ampersand) al parámetro que se desea pasar por referencia.

1.5.11 ARRAYS

Un array es un conjunto de elementos de un tipo definido (colección de datos homogéneos) que puede contener "n" elementos. Por ejemplo, uso un array, para almacenar nombres, asignando secuencialmente a la variable **$agenda[]** (que es nuestro array) los nombres que se desea almacenar.

```
$agenda[] = 'Jorge';
$agenda[] = 'Rodrigo';
$agenda[] = 'Jonathan';
```

Otra forma de asignar valores al array es indicando la posición. El array hasta el momento tiene los siguientes valores Jorge, Rodrigo y Jonathan.

Por defecto, en un array para hacer referencia al primer elemento es, utilizado el índice **$nombre_array[0]**, aunque se puede utilizar cualquier valor. Este es un punto a destacar porque a veces se pueden tener errores en la programación al pensar que la primera posición es la 1.

Un array es un número de elementos, donde cada elemento puede almacenar un valor. Estos elementos están referenciados por una clave que permite acceder a ese elemento.

El orden de los números para el índice del array no es estrictamente necesario ya que podríamos definir los valores dentro de cualquier posición.

También se podría inicializar un array, utilizando el constructor **array**, como se verá en el siguiente ejemplo.

```
<?
     $mi_array =
array("Argentina","Uruguay","Mexico","Chile","Bolivia");
     //Muestro Mexico
     Echo $mi_array[2];
?>
```

Recorriendo Arrays.

Como se ha visto existen diversas formas para inicializar una array dentro del código PHP y como contar la cantidad de elementos que tiene cada uno. Ahora bien, si se combinan ambas cosas y un bucle como el for, se podrá comenzar a recorrer un array dinámicamente, para que muestre todos los elementos que lo componen.

Arrays secuenciales

Los arrays secuenciales son aquellos en los cuales los elementos han sido cargados en forma secuencial y están en posiciones contiguas.

Arrays no Secuenciales

Existen funciones para moverse dentro de los arrays que son **next()** y **prev().** La función **next()** mueve el puntero interno a la posición siguiente. Si no hay más elementos en el array, porque se llega al final de este, devolverá un valor falso. Si el array contiene elementos vacíos, también devolverá falso. La función **prev()** mueve el puntero anterior. Si no hay más elementos también devuelve un valor falso, al igual que si el array tiene elementos vacíos

Arrays multidimensionales

En un array se puede almacenar elementos de distintos tipos: Esto significa que se podrá guardar también de un array otros arrays: Si dentro de un array se guarda otro array, se tendrá como resultado un array multidimensional.

1.5.12 MANEJO DE ARCHIVOS

Resulta de mucha utilidad poder guardar datos en archivos, datos que en caso contrario se perderían al cerrar el navegador. También se piensa en todas las posibilidades que tendríamos al poder subir archivos al servidor mediante un formulario.

Cuando se va a realizar cualquier operación con un archivo nuevo o ya existente, se debe utilizar tres operaciones básicas.

1. Abrir el archivo
2. Modificar el contenido
3. Cerrar el archivo

Abrir archivos

La función que se debe utilizar para abrir una archivo es **fopen()**.

La sintaxis es la siguiente:
fopen(nombre_archivo, modo)

- El parámetro **nombre de archivo** es simplemente el nombre del archivo que deseamos abrir, o crear si no existe.
- El parámetro **modo** corresponde a la forma de apertura del archivo, de la cuál dependerá el tipo de operación que se desea realizar con él.

Algunas funciones para archivos

fpassthru().- Función para ver el contenido de un archivo, que recibe como parámetro el puntero del archivo que se quiere leer.

fread().- Permite leer parte de un archivo abierto.

fgetc().- Permite obtener un carácter del puntero al archivo que le pasamos como parámetro. Esta función devuelve como resultado falso en caso de llegar al final del archivo.

fgets().-También permite leer una cadena de texto.

file().- Permite almacenar cada línea del archivo como un elemento de un array, siendo la primera línea el elemento cero del array.

fputs().- Esta función permite escribir en un archivo. Esta recibe tres parámetros, de los cuales 2 son obligatorios y el tercero opcional.

- El primer parámetro es el puntero al archivo.
- El segundo parámetro es el texto que se desea escribir en el archivo.
- El tercero es el largo de la cadena. Se le omite, la cadena será escrita.

fwrite().- Es exactamente igual a **fputs()**. En realidad **fputs()** es un alias de **fwrite()**.

PHP provee distintas instrucciones para incluir el contenido de un archivo dentro de otro archivo. Esto será de suma utilidad, ya que se podrá crear, por ejemplo un archivo de funciones donde se incluiran todas las funciones de nuestra aplicación. Luego se podrá llamar al archivo de funciones para tener disponibles todas las funciones incluidas en el archivo.

Require().- La instrucción **require()** se reemplaza por el contenido del archivo que recibe como parámetros. Se usa principalmente para hacer referencia a funciones, constantes y otras que no van a ser modificadas en la aplicación Web, por ejemplo los valores para conectarse a una base de datos.

Include().- Esta función difiere de **require()** en los tipos de errores que producen. La principal diferencia entre **require** e **include** es que **require**, reemplaza su llamada por el contenido del fichero requerido, e **include,** incluye y evalúa el fichero especificado.

Otra diferencia entre **require** e **include** es que si hacemos un **require** de un archivo que no existe, nos dará dos errores, primero un Warning, y luego un Error Fatal, que detendrá la ejecución del programa. Mientras que si hacemos un **include** de un archivo que no existe, sólo se recibirá un *Warning que no cancelará la ejecución en curso.*

1.5.13 PASAJE DE VARIABLES A TRAVÉS DE PHP

Existen dos métodos para pasar los valores de las variables a los servidores.

- Método Get
- Método Post

Cuando se utiliza el método GET, todos los valores son pasados en una URL y se ven como algo similar a esto:
http://www.miservidor.com/programas/formulario.php?nombre=Martin

Esta dirección ejecutará el programa denominado formulario.php en la carpeta de programas en el servidor **miservidor.com.** También pasa una variable llamada nombre a la cuál le asigna el valor "Martin".

Cuando en una dirección Web se pone el símbolo **?** se esta indicando que a partir de ahí comenzara a pasar parámetros.

1.5.14 COOKIES

Todos los servicios webs hacen un rastreo básico sobre todos los elementos que se solicitan y los accesos a ellos. Toda esa información se almacena en un registro llamado log, donde figura el archivo solicitado, la fecha, la dirección ip, entre otras características. Contener toda esta información resulta de utilidad para ver las estadísticas del sitio Web, pero no nos permite saber con exactitud si un usuario navegó por alguna página determinada, si completó o no algún formulario, si ya visitó una determinada página, cuantas veces lo hizo, etc. Para poder solucionar estos problemas se utilizan las cookies, que son archivos de texto que son almacenados en el browser del usuario que recorre el sitio.

Las cookies son pequeños archivos de texto generados por el servidor y que van a estar almacenados en el navegador del usuario que está visitando nuestro sitio Web, para que luego puedan ser leídos por nuestro servidor.

Se pueden crear cookies utilizando la función **setcookie(),** cuya sintaxis es la siguiente:

```
setcookie(nombre, valor, expiración, path, dominio, secure)
```

Para borrar una cookie, utilizamos la misma que función para crearla, con la diferencia que no especificaremos ningún valor.

Las limitaciones de una cookie es sumamente importante, porque, por ejemplo, si se crea una cookie dentro de un directorio /directorio1/, la cookie tendrá validez solamente para ese directorio y todos sus subdirectorios. Por lo tanto si queremos utilizar también en /directorio2/ no se puede. De esta manera, se podrá limitar el acceso a determinados directorios o programas.

Un último parámetro que será de utilidad para limitar el alcance de una cookie es el parámetro secure. Para especificar que la cookie debe ser segura, se debe poner el parámetro secure.

Las sesiones permiten monitorear cuales son las páginas que visita un usuario en nuestro sitio Web, de esta manera el uso de sesiones será fundamental en aquellos sitios webs donde se ofrecen servicios de personalización o es necesaria la identificación de usuarios a través de datos sensibles.

1.6 INTERACCIÓN DE WWW CON BASES DE DATOS

Una base de datos es un repositorio de datos, el cual esta compuesto por tablas. A su vez, cada tabla está compuesta por registros (fields) y cada campo tiene propiedades, por ejemplo el tipo de dato que puede almacenar. El conjunto de campos de una tabla constituye un registro. Una base de datos nos permitirá agregar funcionalidades a nuestro sitio Web y podrá empezar a interactuar con él de una manera mucho más eficiente. La gran mayoría de los sitios Web, que generalmente manejan un volumen de información importante, utilizan bases de datos para poder almacenar la información y modificarla o borrarla en tiempo real.

1.6.1 TIPOS DE BASES DE DATOS

Hay muchos tipos de bases de datos en el mercado, de distintos, fabricantes, con algunas diferencias en funcionalidades, u otros aspectos.
Por mencionar algunas podemos destacar.

- Oracle
- MySQL
- MS SQL
- Access
- PostgreSQL

También se tienen diferentes tipos de bases como se verá a continuación, aunque solo se harán algunos comentarios al respecto...

1.6.2 INTRODUCCIÓN A SQL

La interacción con la base de datos se realiza a través del lenguaje SQL (Structured Query Language) que es el estándar para acceso y manipulación de información de una base de datos relacional. SQL es un lenguaje estándar y es soportado por la mayoría de las bases de datos relacionales, aunque cada base tiene su particularidad.

La estructura básica de una expresión SQL consiste en tres cláusulas: **select**, **from** y **where.**

Una consulta modelo de selección de los registros en una tabla sería:

```
Select * from nombre_tabla
```

Esta consulta significa que selecciona todos los campos (mediante **select ***) de la tabla que ubiquemos en nombre_tabla (from **nombre_tabla**).

Algunos motores de base de datos pueden llegar a tener algunos inconvenientes. PostgreSQL no se ve afectado por las mayúsculas o minúsculas en nada de su sintaxis.

```
 sEleCt nombre telefono, direccion from contactos
selEct nombre, telEFONO, direCCion from contactos
```

Ambas consultas (querys), producen los mismos resultados.

Querys

Un Query es una consulta que se le pasa a una base de datos para que ésta ejecute dicha consulta o Query. Los querys se escriben en el lenguaje SQL.

Por ejemplo, si se tuviera un programa de una agenda, que permitiera guardar contactos y teléfonos de los mismos en una base de datos, cuando se busca algún teléfono o nombre, lo que estaríamos haciendo es un query (una consulta) a la base de datos de dicho programa para que nos trajera el número de teléfono del contacto que se esta buscando y el nombre de dicho contacto

1.6.3 CREACIÓN DE LA BASE DE DATOS

Es la primera operación que se debe realizar para crear la base de datos. La sintaxis para crear la es la siguiente.

```
CREATE DATABASE agenda;
```

Con esto ya se ha creado la base de datos de agenda, pero aún falta definir todo el contenido de ésta (tablas y campos).

1.6.4 BORRADO DE LA BASE DE DATOS

Esta operación permite borrar la base de datos, junto con todas las tablas y registros contenidos en ella. Para borrar una base de datos se utiliza el comando **DROP DATABASE**. Por ejemplo, si se quisiera borrar la base de datos agenda que se crea previamente usaríamos esta sintaxis.

```
DROP DATABASE agenda;
```

1.6.5 CREAR UNA TABLA

Creada la base de datos agenda, ahora se crea una tabla dentro de ésta para poder guardar los registros de los contactos. Se utiliza la siguiente sintaxis:

```
CREATE TABLE nombre_de_la_tabla_a_crear
```

1.6.6 BORRAR UNA TABLA

Si se quiere borrar una tabla y todos los registros e índices que ella contiene se utiliza el comando `DROP TABLE.`

La sintaxis es la siguiente:

```
DROP TABLE nombre_de_la_tabla_a_borrar
```

1.6.7 ALTA DE UN REGISTRO

Cuando se quiere insertar un registro, se utiliza INSERT.

INSERT

Esta sentencia permite ingresar nuevos datos en tablas de la base. Su sintaxis es la siguiente:

```
INSERT INTO tabla ([CAMPO1, CAMPO2, …]) VALUES ([VALOR1,VALOR2,
...]);
```

Tabla corresponde a la tabla en la cual se quiere añadir un registro. `CAMPO1, CAMPO2`, etc... son los campos a los cuales se quiere asignar un valor para el registro añadido recientemente. Es importante notar que los valores se asignan a los campos según el orden en el que éstos son nombrados en los paréntesis siguientes del nombre de la tabla, de esta forma, `VALOR1` se asignará a `CAMPO1, VALOR2 a CAMPO2,` etc...

1.6.8 CONSULTAS SOBRE LOS REGISTROS EXISTENTES EN UNA TABLA

Siempre se guardan los datos en una tabla para luego poder consultarlos, ya que si no se pudieran consultar no tendría mucho sentido todo lo que estamos haciendo.

SELECT

Esta es quizás la sentencia más importante del lenguaje SQL[5]. Es la que permite obtener datos almacenados en la base, permitiendo a su vez ordenarlos y filtrarlos según criterios sumamente elaborados. La sintaxis es básicamente la siguiente:

```
SELECT [ CAMPO | * ] FROM [TABLA];
```

Donde `CAMPO` puede ser una lista de campos separados por comas.
El resultado de ejecutar esta sentencia es una tabla que contiene la información de todos los registros almacenados en la tabla `TABLA` pero únicamente aquellos campos seleccionados.

Para filtrar registros (seleccionar sólo una parte de los registros de una tabla) se utiliza la palabra clave `WHERE` de la siguiente manera:

```
SELECT * FROM TABLA WHERE CAMPO > 5;
```

De esta forma se obtiene una tabla con todos los registros de la tabla `TABLA` pero solamente de aquellos que cumplan con la condición de que el dato contenido en su campo `CAMPO` sea mayor que 5. En la cláusula `WHERE` pueden ser expresadas condiciones sumamente complejas, incluyendo todas las operaciones booleanas, lo cual constituye uno de los puntos más fuertes del lenguaje SQL.

[5] Es importante notar que la sintaxis de SQL es común para todo producto de bases de datos que adhiera al Estandard.

1.6.9 ORDENANDO LOS RESULTADOS

Si se tuviera muchos registros, sería de mucha utilidad listarlos ordenados en forma alfabética. Para lograr este listado ordenado se utilizan las palabras claves `ORDER BY:`

```
SELECT * FROM TABLA WHERE CAMPO < 7 ORDER BY CAMPO2;
```

Nuevamente, esta sentencia obtiene una tabla compuesta por todos los registros de la tabla que cumplan con la condición de que su campo sea menor que 7 pero ordenados en forma ascendente por el campo `CAMPO2`.

1.6.10 BORRANDO UN REGISTRO

Cuando se quiere borrar un registro previamente ingresado para darlo de baja se utiliza la instrucción DELETE

DELETE

Esta sentencia permite eliminar datos existentes en tablas de la base. Su sintaxis es la siguiente:

```
DELETE FROM [TABLA] WHERE [CONDICION];
```

Donde `TABLA` es la tabla de la cual se quieren eliminar filas y `CONDICION` es el criterio que se utilizará para decidir cuáles de ellas deben ser eliminadas. Es importante especificar un criterio, dado que de otra forma se asume que el criterio es cumplido por **todos** los registros, lo cual daría como resultado la eliminación total de los registros de la tabla.

1.6.11 EDICIÓN DE UN REGISTRO

Cuando se quieren modificar un registro para actualizar alguno de sus campos se utiliza la instrucción UPDATE.

UPDATE

Esta sentencia permite actualizar datos existentes en tablas de la base. Su sintaxis es la siguiente:

```
UPDATE [TABLA] SET [CAMPO1]=[VALOR1], [CAMPO2]=[VALOR2] WHERE
[CONDICION];
```

Donde TABLA es la tabla de la cual se quiere actualizar filas y CONDICION es el criterio que se utilizará para decidir cuáles de ellas deben ser actualizadas. Es importante especificar un criterio, dado que de otra forma se asume que el criterio es cumplido por **todos** los registros, lo cual daría como resultado la actualización de todos los registros de la tabla.

Clave primaria

La clave primaria es sumamente importante, ya que permitirá referirse de manera unívoca a cada uno de los registros que se encuentran en una tabla. Por ejemplo, cada cliente es una empresa podrá estar identificado por un número a través del cual se podra obtener toda clase de información relacionada con el mismo.

1.6.12 ¿QUÉ ES MYSQL?

MySql es un servidor de bases de datos relaciones (RDBMS). Sus características principales son:[6]

- Alta velocidad
- Alta confiabilidad
- Sencillez
- Código abierto

Dado que MySql es un producto de código abierto y está desarrollado principalmente para la plataforma Linux, se ha convertido en el compañero ideal de PHP.

MySql, a diferencia de otros productos de administración de bases de datos, no viene con una interfaz gráfica incorporada, sin embargo, existe un intérprete de comandos que permite la interacción con el servidor.

MySql utiliza la sintaxis Standard ANSI-SQL para la manipulación de las bases de datos.

¿Qué se puede hacer con MySql?

MySql es un excelente producto para el almacenamiento seguro de grandes volúmenes de datos complejos. Cualquier aplicación que requiera soporte de bases de datos puede utilizar MySql. Particularmente su uso para aplicaciones de Internet está dado por su velocidad de respuesta frente a otras alternativas comerciales.

Como todo servidor de bases de datos, MySql utiliza una estructura de seguridad para acceder a los datos almacenados, por lo tanto, siempre que se desee conectar al servidor se debe identificar mediante un nombre de usuario y una contraseña.

Funciones de PHP para interactuar con MySQL

PHP permite conectar con muchas otras bases de datos a través de una API o una conexión ODBC, aunque lo que ahora interesa es MySQL.

PHP provee diferentes funciones según el tipo de base de datos con la cual se va a trabajar, como:

[6] Minera José Francisco, PHP y MySQL, (2005), p 71

Ora_logon() –función de oracle
Sybase_connect() – función de sybase
ODBC_connect() – función de ODBC
Pg_connect() – función de PostgreSQL
Ifx_connect() – función de Informix

Para trabajar con MYSQL y PHP, se verá a continuación las principales funciones (Tabla 6.1):

Tabla 6.1 Principales funciones acceso a MySQL desde PHP

Función	*Definición*
Mysql_connect ()	Esta función abre una conexión con el servidor MYSQL. Esta retorna un identificador que luego será utilizado por otras funciones MYSQL. El llamado a esta función debe realizarse previamente a cualquier otra función Mysql. La sintaxis es la siguiente: mysql_connect ("servidor", "usuario", "contraseña")
Mysql_close()	Esta función cierra una conexión a la base de datos MySQL abierta anteriormente. Esta recibe como parámetro optativo el identificador de la conexión. En caso de que este identificador no esté definido, la función cerrará la última conexión abierta.
mysql_select_db()	A través de esta función se podrá seleccionar luego de la conexión con el servidor, con que base de datos a trabajar. Es similar a usar nombre_base_de_datos desde la línea de comandos de MySQL. La sintaxis de esta función es la siguiente: mysql_select_db(nombre_base_de_datos, id_conexion)
Mysql_num_rows( )	La función mysql_num_rows() nos permite conocer el número de registros obtenidos de una consulta efectuada con mysql_query().
Mysql_query()	Esta función permite enviar una instrucción SQL al servidor Mysql para que sea ejecutada. Su sintaxis es la siguiente.
Mysql_query(instrucción sql, id_conexion)	Según el código entonces la función recibe como parámetros la instrucción sql que se desea ejecutar y el identificador de la conexión a la base de datos. Si se omite colocar el identificador, utilizará la última conexión abierta.
Mysql_result()	Esta función permite obtener el dato en función del resultado de la consulta. La sintaxis correcta para emplear esta función es la siguiente. mysql_result(identificador_mysql_query, columna,campo)
Mysql_fetch_array( )	Esta función devuelve el contenido de una consulta Select en un array identificando cada elemento con un campo de la base de datos. Esta recibe como parámetros el identificador que se utiliza al ejecutar la función. **mysql_query()**

Función	*Definición*
Mysql_field_type()	Con esta función se podrá obtener el tipo de datos que almacena un campo en la base de datos. La sintaxis es muy parecida a la función anterior, ya que también recibe como parámetros el identificador de la consulta y el índice del campo
Mysql_errno()	La función mysql_errno() nos devolverá el número de error de la última ejecución solicitada al servidor mysql. Si el resultado es un cero, significa que no ha ocurrido ningún error.
mysql_error()	Informa el mensaje de error más reciente que se haya producido en el servidor mysql.
Mysql_create_db()	Esta función permite crear una nueva base de datos en el servidor mysql luego de haber realizado una conexión previamente
Mysql_drop_db()	La función inversa a la anterior y que se utiliza de forma idéntica, la cual nos permite eliminar una base de datos. Recibe como argumentos el nombre de la base de datos que deseo borrar y el identificador de la conexión al servidor
Mysql_free_result()	Esta función libera el resultado. Se llama a esta función por si preocupa utilizar demasiada memoria durante la ejecución de un Script.

1.7 INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

La seguridad en cómputo es un conjunto de recursos destinados a lograr que los activos de cómputo, redes e información de una organización sean confidenciales, íntegros y disponibles para los usuarios. La seguridad informática se logra aplicando un conjunto de reglas, procedimientos y actos tendientes para salvaguardar la información contenida en los sistemas de una empresa.[7]

1.7.1 HISTORIA DE LA SEGURIDAD.

La historia de la seguridad informática se remonta a los tiempos de los primeros documentos escritos. La necesidad de información segura tuvo su origen en el año 2000 antes de Cristo. Los egipcios fueron los primeros en utilizar jeroglíficos especiales para codificar la información y, según paso el tiempo, las civilizaciones de Babilonia, Mesopotamia y Grecia inventaron formas de proteger su información escrita. La codificación de la información, que es el base del cifrado, fue utilizada por Julio César, y durante toda la historia en periodos de guerra, incluyendo las guerras civiles y revolucionarias, así como en las dos guerras mundiales. Una de las máquinas de codificación mejor conocidas fue la alemana Enigma, utilizada por los alemanes para crear mensajes codificados en la Segunda Guerra Mundial. Con el tiempo, y gracias a los esfuerzos del proyecto Ultra de los Estados Unidos, entre otros, la capacidad de descifrar los mensajes generados por los alemanes marcó un éxito importante para los aliados.

En los últimos diez años, la importancia de la seguridad informática se ha puesto de manifiesto por algunas historias. Una de ellas fue la del gusano de Internet, en 1988, que se extendió por decenas de miles de computadoras, como resultado de la obra de un hacker llamado Robert Morris. Había un pirata informático en 1995 en Alemania que se introdujo en casi 30 sistemas a partir de un objetivo que se había propuesto así mismo de casi 500. En febrero de 1995, el arresto del pirata informático más buscado, Kevin Nitnick, reveló las actividades criminales que incluían el robo de códigos, de información y de otro tipo de datos secretos durante años. Claramente, la amplia utilización de los sistemas informáticos ha puesto en evidencia la importancia de la seguridad informática. El objetivo principal de ésta es proteger los recursos informáticos del daño, la alteración, el robo y la pérdida. Esto incluye los equipos, los medios de almacenamiento, el software, y en general, los datos.

1.7.2 CONCEPTOS DE SEGURIDAD.

Autentificación.

La autenticación se refiere a demostrar la identidad de las entidades involucradas en la transacción. Evita que alguien tome la identidad de otro. Generalmente toma dos formas:[8]

1. Autenticación del proveedor de bienes o servicios
2. Autentificación del cliente

[7] Petersen Richard, Manual de referencia Linux, (2000), p 1057
[8] Petersen Richard, Manual de referencia Linux, (2000), p 1064

Un entorno operativo seguro exige la garantía de que sólo puedan acceder a una determinada información aquellos que están autorizados para ello, que la información se procese correctamente y que está disponible cuando se necesita.

Para aplicar controles adecuados, es preciso comprender primero quién o qué es lo que amenaza dicho entorno, así como conocer los riesgos asociados a dichas amenazas si llegan a materializarse. La autenticación pretende establecer quién eres. La autorización (o control de acceso) establece qué puedes hacer con el sistema.

Confidencialidad.

La confidencialidad tiene relación con la protección de información frente a posibles accesos no autorizados, con independencia del lugar en que reside la información o la forma en que se almacena.

Disponibilidad.

Un sistema posee la propiedad de disponibilidad, sí la información es accesible (está disponible) en el momento en que así lo deseen los usuarios, entidades o procesos autorizados.

La disponibilidad es la garantía de que los usuarios autorizados puedan acceder a la información y recursos cuando los necesiten. La falta de disponibilidad se manifiesta principalmente de dos formas:

Integridad.

Un sistema posee la propiedad de integridad si los datos manipulados por éste no son alterados o destruido por usuarios, entidades o procesos no autorizados.

La **integridad** se refiere a la protección de información, datos, sistemas y otros activos informáticos contra cambios o alteraciones en su estructura o contenido ya sean intencionados, no autorizados o casuales.

También es importante proteger los procesos o programas que se emplean para manipular los datos. La información se debe preservar y poner a disposición de sus propietarios y de los usuarios autorizados de una forma precisa, completa y oportuna.

No repudio.

Permite comprobar las acciones realizadas por el origen o destino de los datos.

- Con pruebas de origen.
- Con pruebas de entrega.

Los mecanismos principales son los certificados, la notarización las firmas digitales. Con el esto se logra de alguna manera garantizar que alguien que hay recibido un pago no pueda negar este hecho, de tal forma que es necesario garantizar que alguien que haya efectuado un pago no pueda negar haberlo hecho.

Tipos de seguridad.

Hay dos tipos de seguridad, que comprenden dos universos distintos de una misma plataforma: la seguridad **lógica** y la **física**. La primera se refiere a los datos, al acceso de personas no

autorizadas a la información privada. La segunda tiene que ver con el acceso a donde se encuentren alojados los equipos, sistemas de comunicaciones y demás.

1.7.3 CONTROL DE ACCESO

Nombres de usuarios.

Cada persona que va a tener acceso a un sistema necesita un nombre de usuario para poder autenticarse ante él. Los nombres de usuarios pueden ser cualquiera, de igual manera es tan importante que una contraseña, pues uno depende del otro. Es recomendable que el nombre de usuario sea fácil de recordar, así como su contraseña. Tal vez no haya una regla específica para asignar un nombre de usuario a alguna persona o usuario, pero si hay que tener en cuenta que el nombre no sea significativo al nombre de usuario de algún administrador del sistema.

Grupos de usuarios.

Para tener una mejor administración sobre los usuarios que acceden a nuestros sistemas, se forman grupos de trabajo, a los cuales se asignan a los usuarios. De esta forma se sabe, incluso, que tipo de usuario es, a que sistemas puede acceder, y que permisos tiene en el esquema. Un usuario puede pertenecer a uno o más grupos.

Contraseñas (passwords).

Las contraseñas son las claves virtuales para algunos de los recursos más valiosos de la información. Las contraseñas separan lo privado de lo personal para proteger las aplicaciones del escritorio, los buzones de correo electrónico y las cuentas para hacer compras en Internet.

Contraseñas buenas y malas.

Las buenas contraseñas son:

- Únicas. No use una contraseña que ya esté usando en otra cuenta, como por ejemplo el número NIP de su cuenta bancaria.
- Difíciles de adivinar. No use nombres ni lugares comunes.
- De al menos 7 caracteres.
- Una mezcla de mayúsculas, minúsculas, números y símbolos.

Las malas contraseñas son:

- Una palabra completa y correctamente escrita.
- Parecidas a su nombre de usuario o de cuenta (aunque este invertida o duplicada la palabra);
- Nombres comunes, como nombres de familiares, mascotas o amigos.
- Datos de información acerca de usted (por ejemplo las placas de su auto, su número telefónico, número de contribuyente, nombre de su escuela, marca de su auto, nombre de la calle donde vive, etc.);
- Los mismos caracteres repetidos (adivinar estas contraseñas es de lo más fácil para los programas diseñados para ese fin);
- Una secuencia obvia de caracteres (ej. 123456);
- Obvias para cualquiera que le vea escribirlas (por ejemplo "qwerty"... mire cómo están acomodadas esas letras en su teclado y sabrá a qué nos referimos).

¿Cómo elegir contraseñas?

Elija *una contraseña fácil de recordar, pero difícil de adivinar*, que ni su confidente pueda adivinarla.

*Elija una contraseña larga***.** A mayor número de caracteres en su contraseña, más difícil será ésta de adivinar. Cada carácter añadido a su contraseña aumenta el número de combinaciones posibles para descubrirla. Una contraseña larga, pero simple, puede ser tan segura como una corta, pero compleja, y muchas veces más fácil de recordar.

*Use una combinación de letras, números (0-9) y símbolos (! @ # 0 ^ & *)* para que su contraseña sea más difícil de adivinar. Recuerde que su contraseña de Yahoo! sí distingue (y permite) el uso de mayúsculas y minúsculas, lo cual es bueno considerar en el momento en que está creando su contraseña. Una buena técnica es la de elegir una frase para su contraseña. Puede acortarse sustituyendo letras por símbolos o quitando las vocales. Por ejemplo la frase "Todos para uno y uno para todos", pudiera quedar así: "To2pr1y1prTo2".

No use información personal que otras personas puedan adivinar fácilmente, como su cumpleaños, los nombres de sus hijos o su número telefónico. También evite contraseñas como "123456", "prueba" y "contraseña".

Si usa un generador de contraseñas, no comparta ningún dato personal. En Internet existen algunos generadores de contraseñas que le permiten crear una contraseña al azar. Estas contraseñas suelen ser efectivas y difíciles de adivinar, pero también cuesta trabajo recordarlas.

*Mezcle su contraseña, pero consérvela fácil de recordar***.** Trate de sustituir letras con caracteres o con números. También puede quitar vocales o consonantes a esas frases.

1.7.4 SUPER–USUARIOS: ¿QUÉ SON Y PARA QUE SIRVEN?

La cuenta superusuario, normalmente llamada root, viene pre-configurada para facilitar la administración del sistema, y no debería ser utilizada para tareas cotidianas como enviar o recibir correo, exploración general del sistema, o programación.

Esto es así porque el superusuario, a diferencia de las cuentas de usuario, puede operar sin límites, y un mal uso de la cuenta de superusuario puede conllevar desastres. Las cuentas de usuario no pueden destruir el sistema por un error, por ello es generalmente mejor utilizar cuentas de usuario normales cuando sea posible, a no ser que especialmente necesites privilegios extra**.**

1.7.5 ADMINISTRACIÓN BÁSICA DE LA SEGURIDAD

Políticas de cuentas y contraseñas.

Una política de seguridad informática es una forma de comunicarse con los usuarios, ya que las mismas establecen un canal formal de actuación del personal, en relación con los recursos y servicios informáticos de la organización.[9]

[9] Firtman Sebastián, Seguridad Informática (2005), p 84

No se puede considerar que una política de seguridad informática es una descripción técnica de mecanismos, ni una expresión legal que involucre sanciones a conductas de los empleados, es más bien una descripción de lo que se desea proteger y el por qué de ello, pues cada política de seguridad es una invitación a cada uno de sus miembros a reconocer la información como uno de sus principales activos así como, un motor de intercambio y desarrollo en el ámbito de sus negocios. Por tal razón, las políticas de seguridad deben concluir en una posición consciente y vigilante del personal por el uso y limitaciones de los recursos y servicios informáticos

Las Políticas de Seguridad Informática deben considerar principalmente los siguientes elementos:

- Alcance de las políticas, incluyendo facilidades, sistemas y personal sobre la cual aplica.
- Objetivos de la política y descripción clara de los elementos involucrados en su definición.
- Responsabilidades por cada uno de los servicios y recursos informáticos aplicado a todos los niveles de la organización.
- Requerimientos mínimos para configuración de la seguridad de los sistemas que abarca el alcance de la política.
- Definición de violaciones y sanciones por no cumplir con las políticas.
- Responsabilidades de los usuarios con respecto a la información a la que tiene acceso.
- Las políticas de seguridad informática, también deben ofrecer explicaciones comprensibles sobre por qué deben tomarse ciertas decisiones y explicar la importancia de los recursos. Igualmente, deberán establecer las expectativas de la organización en relación con la seguridad y especificar la autoridad responsable de aplicar los correctivos o sanciones.
- Otro punto importante, es que las políticas de seguridad deben redactarse en un lenguaje sencillo y entendible, libre de tecnicismos y términos ambiguos que impidan una comprensión clara de las mismas, claro está sin sacrificar su precisión.

Monitoreo del sistema.

El monitoreo se realiza constantemente sobre las entradas, los procesos y las salidas de los sistemas:

Monitorear los procesos que están ejecutándose de preferencia con *scripts* externos es la mejor manera de ver el estado del sistema Evitar accesos no autorizados al sistema por medio de detectores de intrusos a nivel de host como TRIPWIRE y deshabilitar los servicios que no se estén usando son algunas acciones que se deben llevar a acabo para la administración y del sistema.

Para facilitarnos la tarea, Linux pone a disposición algunos comandos como:

- netsta –an.- Muestra las conexiones punto a punto y los servicios abiertos.
- ps –x.- Muestra los procesos que están corriendo en el sistema.
- top.- Muestra el uso del cpu, y que procesos están consumiendo más recursos.
- md5sum.- Permite comprobar la integridad de archivos y saber si fueron o no alterados.
- Suite **pstools**.- Una poderosa suite de herramientas que permiten monitorear el sistema sin usar los comandos integraos en el servidor, ideal para saber si alguna puerta trasera esta ejecutándose sin consentimiento.

Es incluso de vital importancia revisar las **bitácoras** de acceso al sistema, *syslog*. Se puede encontrar en **/var/log/syslog/**. También es altamente recomendable tener el demonio *syslog* escuchando en una computadora diferente al servidor principal.

1.7.6 SEGURIDAD EN RED

Protocolos de comunicación.

Son el conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red. En este contexto, las entidades de las cuales se habla son programas de computadora o automatismos de otro tipo, tales y como dispositivos electrónicos capaces de interactuar en una red.

Los **protocolos de red** establecen aspectos tales como:

- Las secuencias posibles de mensajes que pueden arribar durante el proceso de la comunicación.
- La sintaxis de los mensajes intercambiados.
- Estrategias para corregir los casos de error.
- Estrategias para asegurar la seguridad.

1.7.7 IMPLICACIONES DE LA SEGURIDAD.

Las principales amenazas o riesgos que enfrentan al utilizan las redes son:

1. **Intercepción de las Comunicaciones**: La comunicación puede ser interceptada y los datos copiados o modificados. La intercepción puede realizarse mediante el acceso físico a las líneas de las redes.

2. **Acceso no Autorizado a computadoras y Redes de computadoras**: Se realiza habitualmente de forma mal intencionada para copiar, modificar o destruir datos. Técnicamente, se conoce como intrusión y adopta varias modalidades: explotación de información interna, ataques aprovechando la tendencia de la gente a utilizar contraseñas previsibles, aprovechar la tendencia de la gente a desvelar información a personas en apariencia fiables e interceptación de contraseñas.

3. **Perturbación de las Redes**: Actualmente las redes se encuentran ampliamente digitalizadas y controladas por computadoras, pero en el pasado la razón de perturbación de la red más frecuente era una falla en el sistema que controla la red y los ataques a las redes estaban dirigidos principalmente a dichas computadoras. En la actualidad, los ataques más peligrosos se concretan a los puntos débiles y más vulnerables de los componentes de las redes como son sistemas operativos, ruteadores, conmutadores, servidores de nombres de dominio, etc.

4. **Ejecución de Programas que Modifican y Destruyen los Datos**: Lamentablemente, los programas pueden usarse también para desactivar una computadora y para borrar o modificar los datos. Cuando esto ocurre en una computadora que forma parte de una red, los efectos de estas

alteraciones pueden tener un alcance considerable. Por ejemplo, un virus es un programa informático mal intencionado que reproduce su propio código que se adhiere, de modo que cuando se ejecuta el programa infectado se activa el código del virus.

5. **Declaración Falsa**: A la hora de efectuar una conexión a la red o de recibir datos, el usuario formula hipótesis sobre la identidad de la computadora que le contesta en función del contexto de la comunicación. Para la red, el mayor riesgo de ataque procede de la gente que conoce el contexto. Por tal razón, las declaraciones falsas de personas físicas o jurídicas pueden causar daños de diversos tipos. como pueden ser transmitir datos confidenciales a personas no autorizadas, rechazo de un contrato, etc.

6. **Accidentes no Provocados**: Numerosos problemas de seguridad se deben a accidentes imprevistos o no provocados como: son tormentas, inundaciones, incendios, terremotos, interrupción del servicio por obras de construcción, defectos de programas y errores humanos o deficiencias de la gestión del usuario, el proveedor de servicio o el usuario.

1.7.8 MECANISMOS DE CONFIANZA.

Firewalls.

Es un sistema o grupo de sistemas que impone una política de seguridad entre la organización de red privada y el Internet. El firewall determina cual de los servicios de red pueden ser accedidos dentro de ésta por los que están fuera, es decir quien puede entrar para utilizar los recursos de red pertenecientes a la organización. Para que un firewall sea efectivo, todo tráfico de información a través del Internet deberá pasar a través del mismo donde podrá ser inspeccionada la información. El firewall podrá únicamente autorizar el paso del tráfico, y el mismo podrá ser inmune a la penetración. Desafortunadamente, este sistema no puede ofrecer protección alguna una vez que el agresor lo traspasa o permanece en el entorno a este.

1.7.9 Políticas de seguridad

Es el conjunto de requisitos definidos por los responsables directos o indirectos de un sistema que indica en términos generales que está y qué no está permitido en el área de seguridad durante la operación general de dicho sistema[10]. Al tratarse de `términos generales', aplicables a situaciones o recursos muy diversos, suele ser necesario refinar los requisitos de la política para convertirlos en indicaciones precisas de que es lo permitido y lo denegado en cierta parte de la operación del sistema, lo que se denomina política de aplicación específica.

Una política de seguridad puede ser prohibitiva, si todo lo que no está expresamente permitido está denegado, o permisiva, si todo lo que no está expresamente prohibido está permitido. Evidentemente la primera aproximación es mucho mejor que la segunda de cara a mantener la seguridad de un sistema; en este caso la política contemplaría todas las actividades que se pueden realizar en los sistemas, y el resto - las no contempladas - serían consideradas ilegales.

[10] Firtman Sebastián, Seguridad informática, (2005), p 103

Ejercicio propuesto Política basado en el RFC 1296

El RFC 1296 trata acerca de listas de dominios y sus servidores.

Cuando se entra en contacto con un servidor (vía el TCP), la primera pregunta es de autorización (SOA) primero se envía para cerciorarse de que el servidor es autoritario para el dominio que es solicitado. Si es así entonces una pregunta de la transferencia de la zona (AXFR) se envía para solicitar todos los expedientes del recurso para el dominio para ser recuperado. Cuando se recibe un expediente del servidor de nombres (NS), el dominio y el servidor referidos se agregan a la lista de dominios al proceso. Cuando se reciben los expedientes del anfitrión (A, CNAME, HINFO, MX), se agregan a en la tabla de la base de la información del anfitrión.

La Política de seguridad describe la forma adecuada de uso de los recursos de un sistema de cómputo, que responsabilidades y derechos tienen los usuarios y administradores y que hacer en caso de incidente de seguridad, entre otras cosas.

La Implementación de la política de seguridad será aplicada al Instituto Mexicano del Petróleo porque es ahí donde estoy realizando mi servicio social.

Política de seguridad basada en el RFC 1296

Todos los usuarios del Instituto pueden acceder al sistema desde cualquier equipo de computo localizado dentro del Instituto o cualquier equipo fuera de este utilizando programas que permitan la conexión segura y sea inmune a sniffing.

Deben autentificarse con su login y password, esta prohibido acceder al sistema con cuentas distintas a la propia.

Si un usuario se conecta desde fuera del Instituto y necesita acceder a la Intranet, pero no dispone de un programa seguro de conexión remota, deberá solicitar que su contraseña sea cambiada temporalmente, ingresar al sistema y solicitar un nuevo cambio de contraseña.

Al momento de ingresar al sistema, cada usuario deberá ser informado acerca de la fecha, hora y máquina desde que se conecto al sistema por última vez, lo cual permitirá detectar fácilmente un uso no autorizado del sistema.

Los usuarios deben de tener un protector de pantalla, el cual se debe de activar después de 10 minutos de inactividad en una Terminal, estación de trabajo o microcomputadora, el sistema automáticamente suspenderá automáticamente la sesión y el usuario sólo podrá reestablecerla proporcionando la contraseña correcta.

Esta prohibido ejecutar programas que intenten adivinar contraseñas alojadas en las tablas de contraseñas de máquinas locales o remotas.

Esta prohibido utiliza programas que intenten detectar las vulnerabilidades de los equipos del Instituto, de forma local o remota.

1.8 DESARROLLO DE APLICACIONES CON POSTGRESQL Y PHP

La POO es una metodología de programación que se fundamenta en el concepto de objeto. Se trata de una filosofía de programación donde la realidad se modela mediante objetos y la interacción entre ellos. Algunas de los beneficios de la de la programación orientada a objetos son:

- Reutilización. Las clases están diseñadas para que se reutilicen en muchos sistemas.
- Estabilidad. Las clases diseñadas para una reutilización repetida se vuelven estables
- El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel.
- Se construyen en clases cada vez más complejas
- Confiabilidad
- Un diseño más rápido
- Integridad
- Migración

1.8.1 LENGUAJES CON SOPORTE A LA POO

Los listados a continuación son ejemplos lenguajes de programación que proporcionan soporte para la programación orientada a objetos:

- Java
- SmallTalk
- Delphi
- Perl
- C++
- C#
- Y evidentemente PHP.

1.8.2 ¿QUÉ ES UN OBJETO?

Un objeto es cualquier cosa real o abstracta, acerca de la que almacenamos datos y lo métodos que controlan dichos datos.

Un objeto es la representación lógica de un elemento físico también se puede definir como la implementación de una clase, es una variable de una clase.

1.8.3 ¿QUÉ ES UNA CLASE?

Una clase es una colección de datos y métodos que operan sobre esos datos. Una clase es una implantación de un tipo de objeto. Especifica una estructura de datos y los métodos operativos permisibles que se aplican a cada uno de sus objetos.

La definición de la clase debe realizarse dentro del mismo bloque de código de PHP

1.8.5 LOS TRES PRINCIPIOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

Todos los lenguajes orientados a objetos proporcionan los mecanismos que ayudan a implementar el modelo orientado a objeto. Estos mecanismos son el encapsulado, la herencia, y el polimorfismo.

Encapsulamiento

El empaquetamiento de datos y métodos se llama *encapsulado*. El encapsulado es el resultado (o acto) de ocultar los detalles de implantación de un objeto respecto de su usuario.

El encapsulado es el mecanismo que permite unir el código junto con los datos que lo manipula, y mantiene a ambos a salvo de las interferencias exteriores y de un uso indebido. Una forma de ver el encapsulado es como un envoltorio protector que impide un acceso arbitrario al código y los datos desde un código exterior al envoltorio. El acceso al código y los datos desde el interior del envoltorio está estrechamente controlado a través de una interfaz correctamente definida.

Herencia

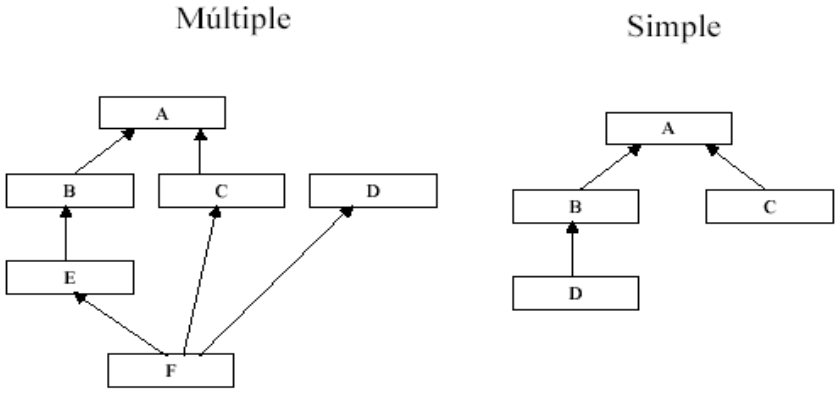
La herencia es el proceso por el cual un objeto adquiere las propiedades de otro. Esto es importante, ya que supone la base del concepto de clasificación jerárquica.

La herencia interactúa también con el encapsulado. Si una determinada clase encapsula determinados atributos, entonces cualquier subclase tendrá los mismos atributos, más cualquiera que añada como parte de su especialización. Esto es un concepto clave que permite a los programas orientados a objetos crecer en complejidad linealmente, en lugar de geométricamente. Una nueva subclase hereda todos los atributos de todos sus predecesores y no tiene interacciones con la mayoría del resto del código del sistema.

La herencia permite el paso de variables y métodos, la reutilización de código, especialización de objetos.

Existen básicamente dos tipos de herencia:

La simple y la múltiple.



La herencia en PHP

La herencia en PHP es simple, no se soporta la herencia múltiple. Para hacer referencia a ella se utiliza la palabra reservada extends en el encabezado de la clase. Se lee 'hereda de'. Se heredan métodos, variables y constructores.

Operaciones con la herencia

Algunas de las operaciones que se pueden hacer con la herencia es la sobre escritura de métodos, utilizado con frecuencia para añadir funcionalidad o para modificar el comportamiento para un método.

Polimorfismo

El polimorfismo (del griego muchas formas) es una característica que permite que una interfaz sea utilizada por una clase general de acciones. La acción especificada queda determinada por la naturaleza exacta de la situación.

1.8.5 OTRAS CARACTERÍSTICAS DE LA POO

Abstracción

Un elemento esencial de la programación orientada a objetos es la abstracción. Los seres humanos abordan la complejidad mediante la abstracción. Por ejemplo, no consideramos un carro como un conjunto de cientos de miles de partes individuales, sino como un objeto correctamente definido y con un comportamiento determinado. Esta abstracción nos permite utilizar el carro para ir al mercado sin estar agobiados por la complejidad de las partes que forman el carro. Podemos ignorar los detalles de cómo funciona el motor, la transmisión o los frenos, y, en su lugar, utilizar libremente el objeto como un todo. En la abstracción se enfatiza en detalles significativos, se suprime lo irrelevante.

Modularidad

Se refiere a fragmentar un programa en componentes para reducir su complejidad en forma de clases, paquetes o bibliotecas, cada uno de ellos con un sentido propio. . Repercute en el acoplamiento o integración de los componentes.

Esta fragmentación disminuye el grado de dificultad del problema al que da respuesta el programa, pues se afronta el problema como un conjunto de problemas de menor dificultad, además de facilitar la comprensión del programa.

Jerarquía

Define una estructura mediante la cual se clasifican los objetos. Permite acceder más fácilmente a la información.

1.8.6 OBJETOS, MÉTODOS Y CONSTRUCTORES

- Los objetos contienen métodos
- El principal de todos los métodos de un objeto se denomina constructor.

- Los constructores sirven para:
  - Inicializar los atributos del objeto.
  - Reservar la memoria necesaria para mantener el objeto.

Constructores en PHP
Previo a PHP 5, requerían tener el mismo nombre que la clase que los contenía.
En PHP 5, se deben llamar __construct() doble guión bajo al inicio del nombre.
En PHP 5, se busca por un método de nombre __construct(), si no se encuentra, se busca con un método con el mismo nombre de la clase, con el fin de mantener compatibilidad con PHP 4.

Firma de un método
- Se denomina así al encabezado de cualquier método.
- Se compone básicamente de los siguientes elementos:
- Tipo de acceso
  - Nombre
  - Parámetros

Sobrecarga de métodos
Se denomina a la cualidad de tener más de un método con el mismo nombre pero firmas distintas. PHP no soporta la sobrecarga de métodos.

"No es valido tener dos métodos con el mismo nombre, aún cuando tengan diferentes firmas"

Referencias a Métodos de la clase base
Al Sobreponer 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 súper clase como se muestra en los siguientes ejemplos:

1.8.7 ATRIBUTOS Y MÉTODOS "STATIC"

El declarar un atributo de forma estática "static", permite que el atributo pueda ser invocado desde fuera de la clase sin la necesidad de instanciar un objeto.

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

1.8.8 VISIBILIDAD

La visibilidad es el nivel de protección para acceder a un atributo o método de una clase.

Public: Pueden ser accesados desde cualquier parte.
Protected: El acceso está limitado a las clases que hijas así como a la clase que define el objeto.
Private: El acceso está limitado sólo a la clase que define el objeto. Privado

1.8.9 INTERFACES

Tipos específicos de clases buscan asegurar que la clase que las implementen integre como parte de su definición determinadas características. Son muy parecidas a la definición de una clase.

- Todos los métodos se declaran sin cuerpo, únicamente va la firma.
- Las clases que utilizan la interfaz deben utilizar la palabra reservada "implements"
- Las clases deben tener un nombre distinto al de las interfaces que implementen.

1.8.10 EXCEPCIONES

PHP 5, integra soporte para el manejo de Excepciones, la intención es tomar en cuenta situaciones anormales dentro de la ejecución del programa y actuar en consecuencia.
Son muy parecidas a la definición de una excepción en Java.
Se utilizan las instrucciones try / catch para capturar las excepciones.
Se utiliza la instrucción throw para lanzar una nueva excepción.
PHP no lanza las excepciones es necesario lanzarlas manualmente.

1.8.11 TEMPLATES

Los programadores suelen ser pésimos diseñadores mientras los diseñadores suelen ser pésimos programadores.

Henry Ford nos mostró las ventajas de la producción en serie, especialistas por área de producción, a mayor experiencia mayor costo.

Los proyectos de Ingeniería de Software, involucran equipos multidisciplinarios, especialistas en cada área, que interactúan constantemente. Y entonces la pregunta es: ¿Cómo se puede separar la capa de Presentación de la funcionalidad y con ello facilitar el proceso de desarrollo?

¿Qué es un Template?

Son plantillas que facilitan el proceso de edición, eliminando el código PHP incrustado, sin eliminar la funcionalidad. Permiten mantener por separado la capa de presentación "Archivos HTML" de la capa de negocios, funcionalidad "Archivos PHP". Esto generalmente se usa cuando son aplicaciones o sitios de gran complejidad y envergadura, permitiendo separar en grupos de trabajo a los programadores y diseñadores.

Ventajas:

- Independencia entre la aplicación y la interfase. Lo recomendado en todo tipo de software.
- Se puede rediseñar un sitio sin tener que cambiar prácticamente nada de código.
- Las actualizaciones a tu sitio serán más fáciles de realizar. Solamente cambias el contenido y no el diseño.
- El mantenimiento del código es más fácil y rápido. No hay que preocuparse por el html.

Desventajas:

- Puede que programar utilizando Templates se torne un poco más pesado. Pero lo Vale.
- El tiempo de procesamiento del Template puede hacer caer el rendimiento de tu sitio. La utilización de un sistema de cache puede solventar la perdida de rendimiento.

Motores de Templates

- "Engines"
- Smarty
- NokTemplate
- FastTemplate
- PHP Savant

Y muchos, en realidad muchos otros. Todos ellos permiten separar la capa lógica de la presentación, con sus diferentes ventajas e inconvenientes.

¿Y como funcionan los Engines de Template ?

La idea fundamental de un "PHP Template Engine" es muy sencilla:

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

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

Lo que hace NokTemplate es interpolación de variables, dada una variable definida en el template, el script lo que hace es intercambiar esa variable por su correspondiente valor (asignado en tiempo de ejecución). Estas variables son expresadas de la siguiente forma: Entre llaves, y los caracteres validos son letras mayúsculas, minúsculas, números y underscore (_). Las variables son sensibles a las mayúsculas y minúsculas.

Noktemplate es muy útil para adquirir las bases y fundamentos de los templates. NokTemplate es relativamente nuevo, fácil de usar y esta en español.

CAPÍTULO 2

PROYECTO DE UN SISTEMA DE CONTROL DE INFORMACIÓN POR INTRANET

CAPÍTULO 2. PROYECTO DE UN SISTEMA DE CONTROL DE INFORMACIÓN POR INTRANET

2.1 OBJETIVO

Implementar un sistema de control de información por intranet, para poder exponer el aspecto práctico del diplomado de desarrollo de sistemas implementados con software libre, dejando a un lado los tecnicismos innecesarios y así de una forma ágil exponer los conocimientos adquiridos en el diplomado.

El objetivo de este trabajo es implementar una mixtura entre dos alternativas: el aspecto teórico y el aspecto practico, para poder exponer de una forma completa todo lo visto en el diplomado séptima edición desarrollo de sistemas implementados con software libre: Linux.

2.2 JUSTIFICACIÓN DEL PROYECTO

Las bases de datos han invadido el ámbito de los sistemas de informáticos desde sus inicios, y se hacen cada vez más notables en nuestra vida cotidiana desde hace ya un tiempo. Este hecho sugiere una necesidad cada vez mas importante (por parte de las empresas) de contar con personal capacitado en el manejo de herramientas que posibiliten el desarrollo rápido de aplicaciones potentes y orientadas tanto a empresas como a particulares, y que, a la vez, impongan el menor costo posible del mercado.

Dadas todas estas situaciones, la elección de la mayoría de las empresas ha sido MySQL, como servidor de base de datos, y PHP como lenguaje de programación de dichas aplicaciones.

El control de información es fundamental para el óptimo desempeño de una empresa.

Diseñar un sistema requiere, en rigor, un análisis previo, que consiste en una serie de pasos, como recopilar información acerca de la empresa en la cual se lo va a utilizar, conocer sus tareas y cómo se las lleva a cabo, cuántas personas trabajan allí, cuáles son las perspectivas y demás. Todas estas actividades son propias de un Análisis de Sistemas.

En este capitulo se implementa el aspecto experimental del diplomado por medio de un sistema de control de información describe la elaboración de un sistema para una intranet para el control de información, inclusive algunos autores consideran el control como una etapa primordial en la administración pues, aunque una empresa cuente con magníficos planes, una estructura organizacional adecuada y una dirección eficiente, el ejecutivo no podrá verificar cuál es la situación real de la organización si no existe un mecanismo que se cerciore e informe si los hechos van de acuerdo con los objetivos.

El concepto de control es muy general y puede ser utilizado en el contexto organizacional para evaluar el desempeño general frente a un plan estratégico.

Una de las razones más evidentes de la importancia del control es porque hasta el mejor de los planes se puede desviar. El control se emplea para:

- Crear mejor calidad: Las fallas del proceso se detectan y el proceso se corrige para eliminar errores.
- Enfrentar el cambio.
- Producir ciclos más rápidos
- Facilitar la delegación y el trabajo en equipo:

Los sistemas de planeación y control son aplicaciones o herramientas de gestión que permiten mejorar y potenciar los recursos de una organización, pueden ser de recursos humanos, nómina, tecnología, servicio al cliente, etc.

La siguiente aplicación ejemplifica un sistema de información para el control de reportes, de un proyecto empresarial.

2.3 DESCRIPCIÓN DEL PROYECTO

Un empleado debe de identificarse, si su login y password son correctos podrá redactar un reporte (resumen) de sus actividades la finalidad es poder detectar cualquier defecto del proyecto en el que se trabaje y así poder corregirlo a tiempo, o simplemente mejorarlo. En caso de que se observe que su proyecto este atrasado o en un mal camino poder replantearlo a tiempo.

Uno de los objetivos de este proyecto es por medio de los reportes almacenados poder llevar un registro de los resultados obtenidos.

Para poder llevar un control adecuado es necesario tener almacenados todos aquellos registros necesarios para poder hacer estadística de los resultados es por eso que este sistema tiene la posibilidad de volver a revisar todos aquellos reportes redactados.

Otra forma de llevar a cabo el control es mediante la validación de los reportes redactados, interpretación y valoración de los resultados. Ese el objetivo del modulo *Validar Reportes* del proyecto.

2.4 SOFTWARE UTILIZADO PARA LA IMPLEMENTACIÓN

Sistema Operativo:	Linux Slackware versión 10.0
Servidor Web:	Apache Web Server versión 2.0
Lenguaje de programación:	PHP versión 5
Base de datos:	MySQL versión 4.1

2.5 CARACTERÍSTICAS DEL CLIENTE

Se puede hacer uso del proyecto desde cualquier equipo que cuente con las siguientes características:

- Conexión a la intranet
- Un navegador Web capaz de interpretar HTML versión 3.2
- Monitor con resolución de 800 x 600 píxeles o superior.

2.6 BASE DE DATOS

El modelo relacional es sin duda el modelo más popular desde hace un tiempo, y con él se imponen conceptos tales como tabla- arreglo bidimensional-, fila y columna. El modelo relacional se entiende como una propuesta de ver los datos como si se trataran de objetos del mundo real, diferentes entre si por sus características básicas. Este modelo admite relaciones uno a varios, uno a uno, y varios a varios. Por todas las características anteriores se eligió este modelo de base de datos además de su funcionalidad y efectividad.

Para poner en práctica los conceptos teóricos vistos, se eligió una base de datos muy popular en este momento y con mucho futuro por delante: MySQL algunas de sus características son las siguientes:

- Rapidez.
- Seguridad.
- Gran estabilidad.
- Administración simple.

Desde hace algún tiempo, se ha ido dando una particular unión entre esta base de datos y el lenguaje de programación PHP; por este motivo MySQL se va a utilizar para el proyecto del Sistema Web.

2.6.1 DIAGRAMA ENTIDAD/RELACIÓN DE LA BASE DE DATOS

La figura 2.1 muestra el diagrama entidad-relación de la bases de datos del proyecto.

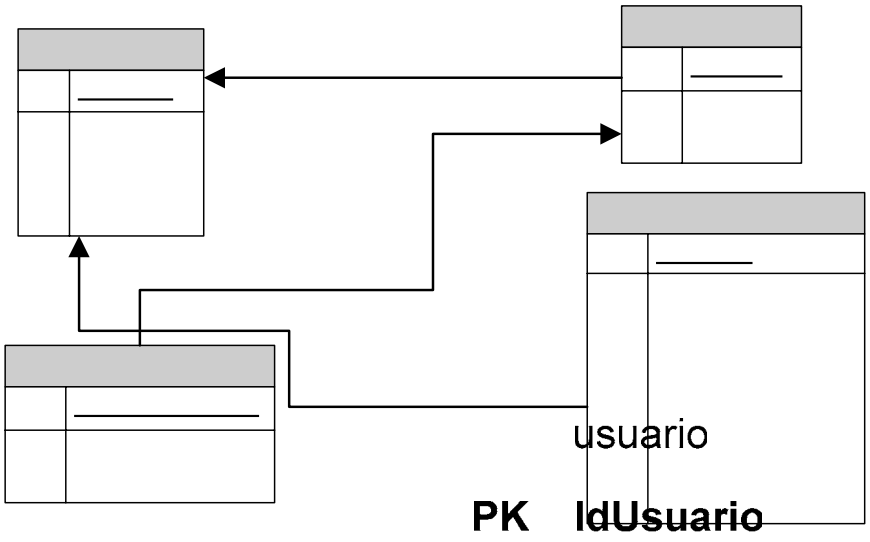


Figura 2.1 Diagrama entidad-relación de la base de datos

2.6.2 DICCIONARIO DE DATOS DE LA BASE DE DATOS

Clasificando por tablas a continuación se muestra el diccionario de la base de datos

```
TABLA usuario
CREATE TABLE usuario(
      IdUsuario int NOT NULL,
      Us_Nombre varchar(20) NOT NULL,
      Us_Apellido varchar(50) NOT NULL,
      Us_RFC varchar(15) NOT NULL,
      Us_Nivel int NOT NULL,
      PRIMARY KEY(IdUsuario)
}ENGINE=innodb;

CREATE TABLE acceso{
      IdAcceso int NOT NULL,
      IdUsuario int NOT NULL,
      Password varchar(20) NOT NULL,
      PRIMARY KEY(IdAcceso),
      INDEX(IdUsuario),
      FOREIGN KEY(IdUsuario)REFERENCES usuario(IdUsuario)
} ENGINE=innodb;

CREATE TABLE reporte{
      IdReporte int NOT NULL,
      IdUsuario int NOT NULL,
      ResumenReporte varchar(200) NOT NULL,
```

```
      Validado int NOT NULL,
      FechaDeElaboracion date NOT NULL,
      Comentario varchar(200) NOT NULL,
      ClaveProyecto int NOT NULL,
      Avance int NOT NULL,
      Actividad varchar(200) NOT NULL,
      Horas int NOT NULL,
      PRIMARY KEY(IdReporte),
      INDEX(IdUsuario),
      FOREIGN KEY(IdUsuario)REFERENCES usuario(IdUsuario)
} ENGINE=innodb;

CREATE TABLE usuario_revisor{
      IdUsuarioARevisar int NOT NULL,
      IdUsuario varchar(20) NOT NULL,
      UsARevisar varchar(20) NOT NULL,
      PRIMARY KEY(IdUsuarioARevisar),
      INDEX(IdUsuario),
      FOREIGN KEY(IdUsuario)REFERENCES usuario(IdUsuario)
} ENGINE=innodb;
```

2.7 ESPECIFICACIÓN DE LOS MÓDULOS DEL PROYECTO

Los reportes que se visualicen del proyecto tendrán el formato que se muestra en la figura 2.2:

- Número de Reporte.- Número del reporte.
- Fecha.- Fecha de elaboración de reporte, con el formato AAAA-MM-DD
- Fecha en la que se elaboró el reporte.
- Clave de Proyecto.- Número de proyecto en el que se esta trabajando.
- Avance.- Porcentaje de la actividad que realiza.
- Horas.- Cantidad de horas que emplea en realizar la actividad.
- Validado.- Si o No son los valores que puede tener este campo.
- Actividad a Realizar.- Titulo de la actividad que realiza.
- Resumen Reporte.- Síntesis de la actividad.
- Comentario del Validador.- Este campo puede estar vació, depende de el validador.

Numero de Reporte	Fecha	Clave Proyecto	Avance	Horas	Validado
8	1995-01-01	212	90 %	3	Si
Actividad a Realizar	Elaboracion de Menu				
Resumen Reporte					
A lo largo de esta semana comenze la elaboracion de el menu prinpicipal, el cual esta implementado con Java Script. Termine el menu, aun falta terminar algunos detalles.					
Comentario del Validador					
Revise el menu, Faltan algunos detalles pero en general quedo muy bien					

Figura 2.2 Formato de los reportes

Este tipo de tabla (Figura 2.3) es común encontrar en el proyecto, es una lista de reportes. Sus campos son los siguientes:

- Numero de Reporte.- Es un enlace del Número del reporte
- Fecha. Fecha de elaboración de reporte, con el formato AAAA-MM-DD
- Validado.- Si o No son los valores que puede tener este campo.

Numero de Reporte	Fecha	Validado
8	1995-01-01	Si
9	1995-01-10	Si
10	1995-01-17	No
11	1995-01-28	No
12	1995-01-30	Si

Figura 2.3 Tabla de Reportes

A continuación se exhibe la funcionalidad del proyecto de control de información.

2.7.1 MENÚ PRINCIPAL

En este menú podemos acceder a uno de los dos posibles módulos (Figura 2.4):

- Validar reporte
- Redactar reporte

En el módulo *Redactar Reporte* (Figura 2.5) un usuario debe identificarse con un login y password mientras que en el módulo *Validar Reporte* (Figura 2.6) se debe elegir un usuario, password y persona a quien se va validar.

En caso de que el login o password sea o sean incorrectos, no se podrá ingresar a los módulos correspondientes. Con color rojo se marca el dato incorrecto, hasta que se introduzca el válido se cargará el módulo que le corresponda (Figura 2.7 y Figura 2.8).

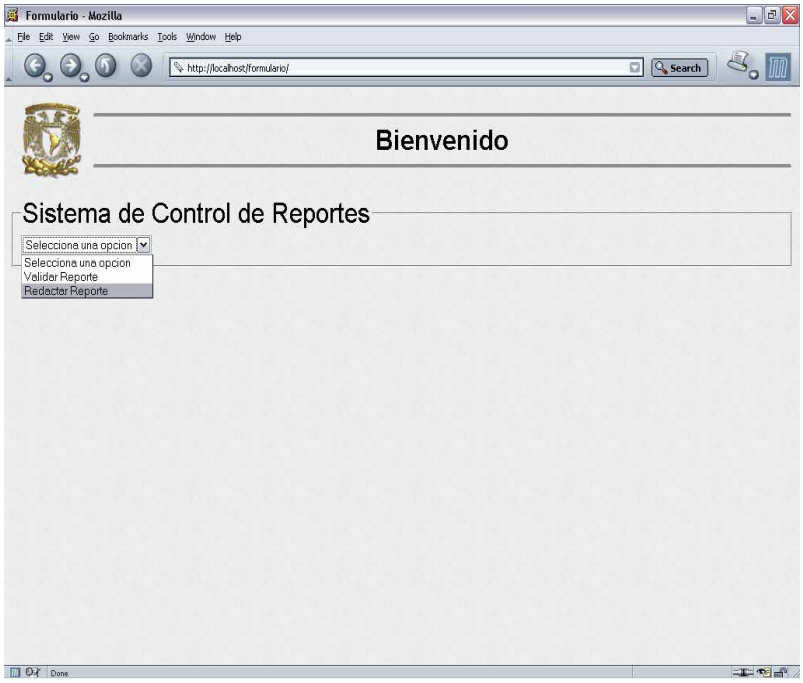


Figura 2.4 Menú Principal

Figura 2.5 Autenticar Redactar Reporte

Figura 2.6 Autenticar Validar Reporte

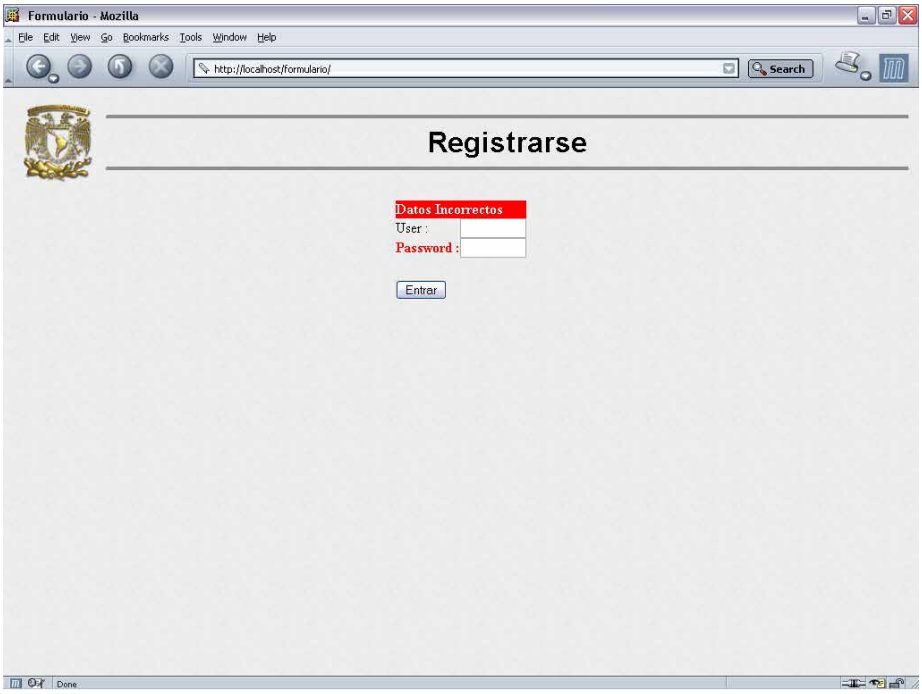


Figura 2.7 Datos incorrectos de Redactar Reporte

Figura 2.8 Datos incorrectos de Validar Reporte

2.7.2 MÓDULO REDACTAR REPORTE

En general en este módulo, el usuario podrá crear, guardar y consultar sus informes (Figura 2.9).

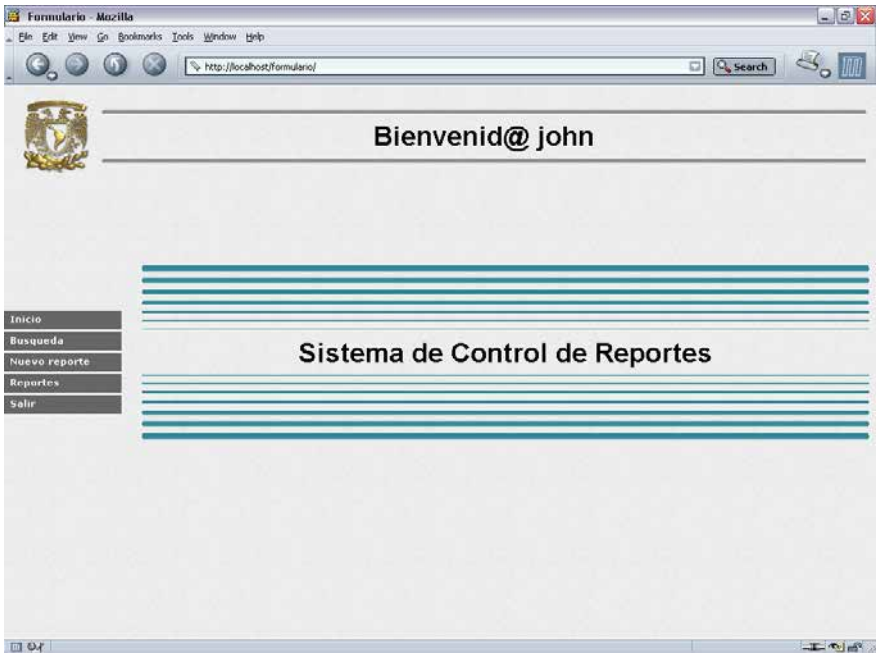


Figura 2.9 Menú principal de Redactar Reporte

Consta de 3 partes

- Búsqueda de reportes. Búsquedas de reportes que están almacenados en la base de datos, dicha búsqueda de reportes se pueden realizar de tres diferentes formas:
  - Por Fecha
  - Por Número
  - Por Clave de proyecto
- Redactar Reporte. Aquí se puede agregar un nuevo reporte
- Reportes. Se muestran los reportes almacenados en la base de datos, con la clasificación .
  - Validados
  - No Validados

2.7.2.1 BÚSQUEDA DE REPORTES

POR FECHA

La búsqueda de reportes *Por Fecha*, consiste en seleccionar de dos listas el mes y el año del reporte que se requiere (Figura 2.10).

El Resultado de la búsqueda consiste en mostrar todos los reportes que se recuperen de la fecha seleccionada (Figura 2.11).

Para poder consultar un reporte en específico de la tabla mostrada en la Figura 2.11 basta con presionar sobre el número del reporte, para que se despliegue en toda la pantalla dicho reporte, y así poder consultar la información (Figura 2.12).

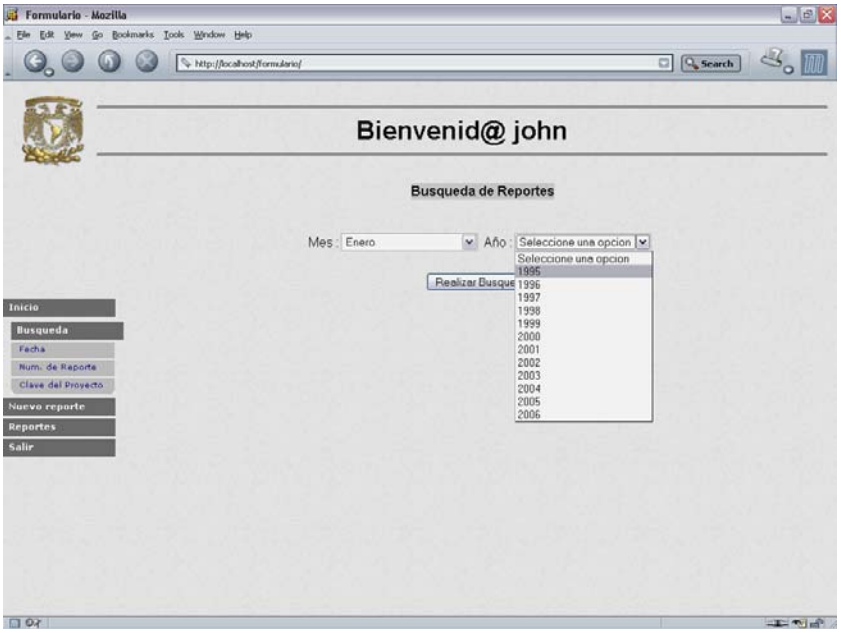


Figura 2.10 Búsqueda de reportes Por Fecha.

Figura 2.11 Resultado de la búsqueda

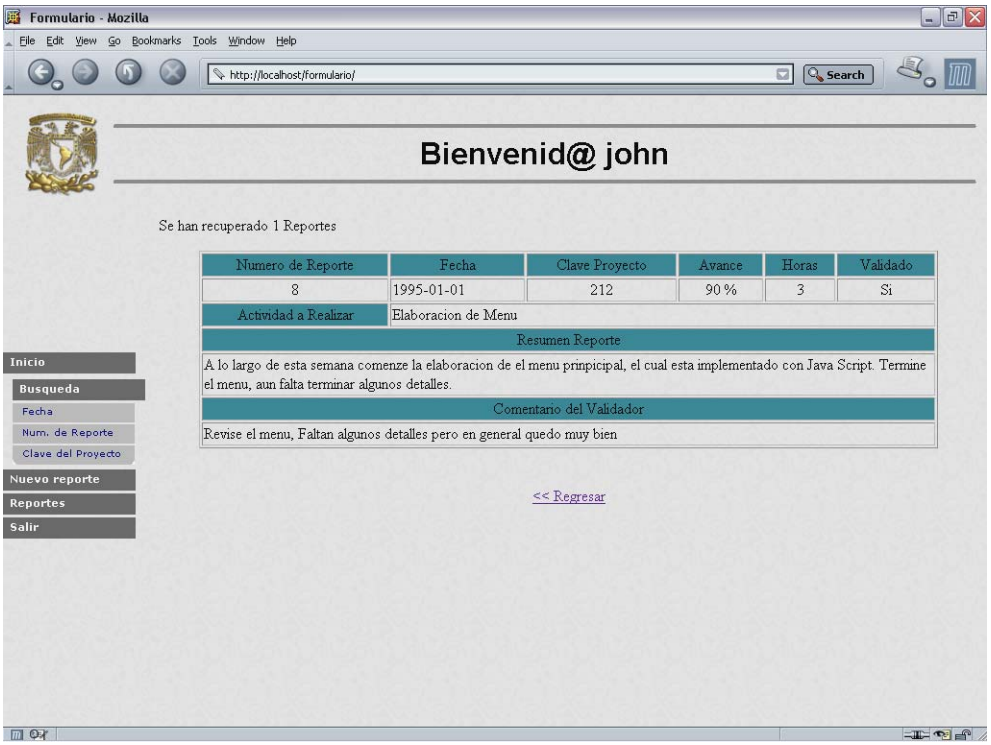


Figura 2.12 Resultado de presionar sobre el reporte numero 8

POR NÚMERO

La búsqueda por número se efectúa, introduciendo en el campo vacío el número de reporte que desea consultar (ver Figura 2.13), basta con presionar sobre el botón *Buscar* para que enseguida se expanda el resultado (Figura 2.14).

Figura 2.13 Búsqueda de reporte por número.

Figura 2.14 Resultado de la búsqueda de reporte número 16

POR CLAVE

La búsqueda por clave se realiza, introduciendo en el campo vació la clave del proyecto de los reportes que desea consultar (ver Figura 2.15), basta con presionar sobre el botón *Buscar* para que enseguida se despliegue el resultado (Figura 2.16).

Figura 2.15 Búsqueda de reportes por Clave de Proyecto

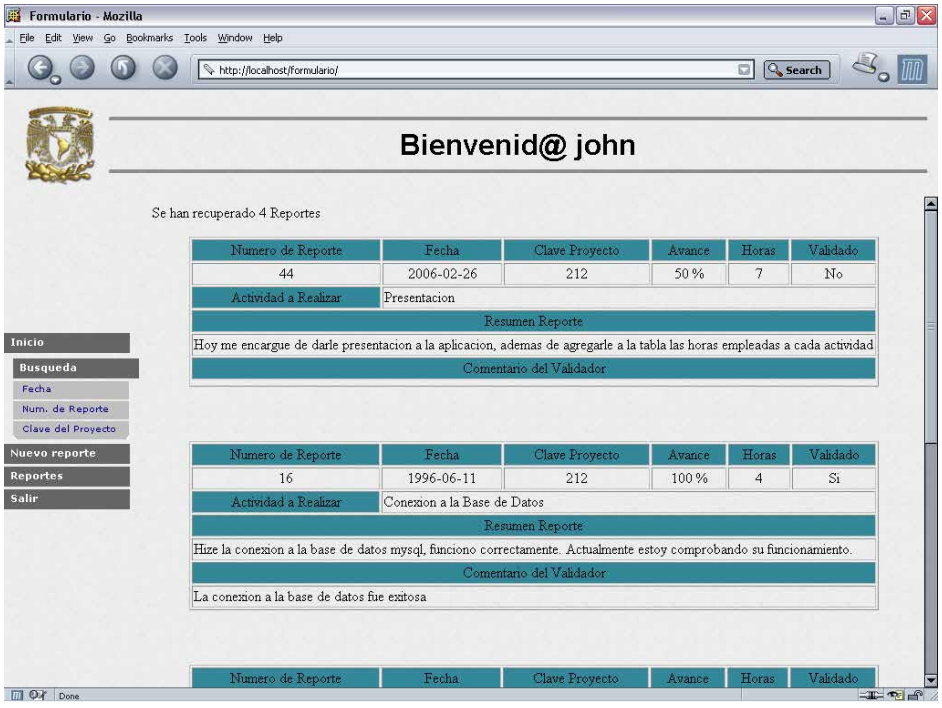


Figura 2.16 Resultado de la búsqueda por Clave de Proyecto 212

2.7.2.2 REDACTAR REPORTE

Nuevo reporte (Figura 2.17) es la sección donde el usuario podrá guardar en la base de datos su informe. El primer paso es presionar sobre *Nuevo reporte* para que se extienda la pantalla donde se va a poder introducir dicho reporte. Los campos ha llenar son los siguientes:

- Clave del Proyecto.- Introducir la clave del proyecto en la que se esta trabajando.
- Actividad a realizar..- Titulo de la actividad que realiza.
- Avance de la actividad..- En porcentaje, escribe la cantidad que ha avanzado en su actividad.
- Horas empleadas a esta actividad.- En número, el total de horas en que trabajo en la actividad.
- Redactar reporte.- Introducir resumen de la actividad.

Como segundo paso, presiona sobre el botón *Enviar reporte* para agregar a la base de datos el reporte.

En caso de que haya redactado mal el reporte, antes de enviarlo puede borrarlo con sólo presionar sobre el botón *Borrar datos*.

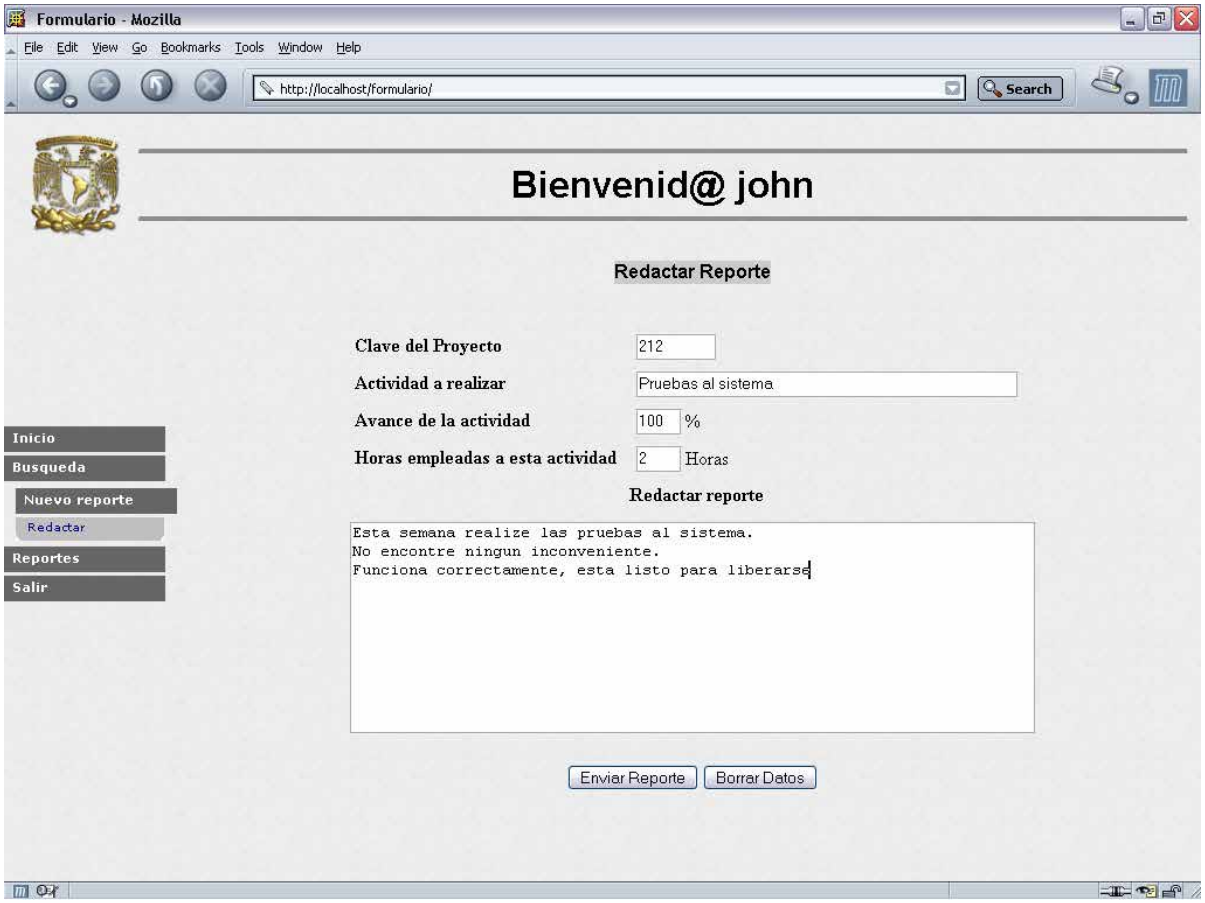


Figura 2.17 Redactar Nuevo Reporte

En la figura 2.18, se ejemplifica el resultado de redactar un reporte correctamente.

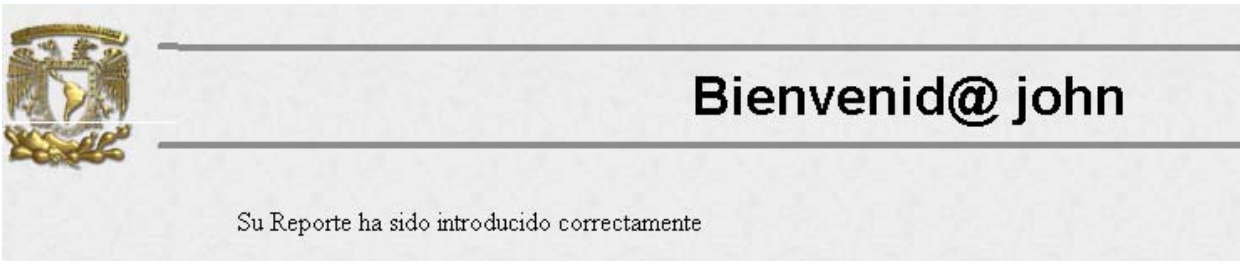


Figura 2.18 Resultado de redactar un nuevo reporte

2.7.2.3 REPORTES

La funcionalidad de esta sección es la de publicar todos los reportes clasificándolos en dos formas:

- Validados
- No validados

VALIDADOS

Con solo dar clic sobre *Validados* expondrá una lista de todos aquellos reportes que ya han sido revisados y validados por el superior. La lista se va publicar en forma de tabla con los siguientes atributos (Figura 2.19):

- Número de reporte
- Fecha
- Clave proyecto
- Validado

Para entrar a un reporte en específico se debe pulsar sobre el enlace. La Figura 2.20 se visualiza el resultado de presionar el enlace del reporte número 8.

NO VALIDADOS

No validados es muy semejante a *Reportes Validados*. La única diferencia es que esta sección como su nombre lo indica exhibe únicamente aquellos reportes que aún no han sido validados, por lo tanto los resultados se obtienen de la misma forma (ver Figura 2.21 y Figura 2.22).

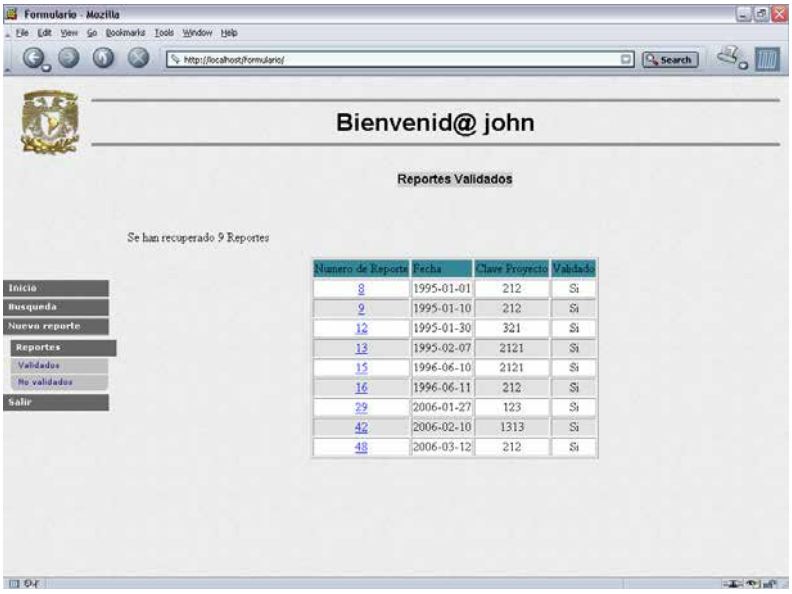


Figura 2.19 Resultado de la búsqueda de Reportes Validados

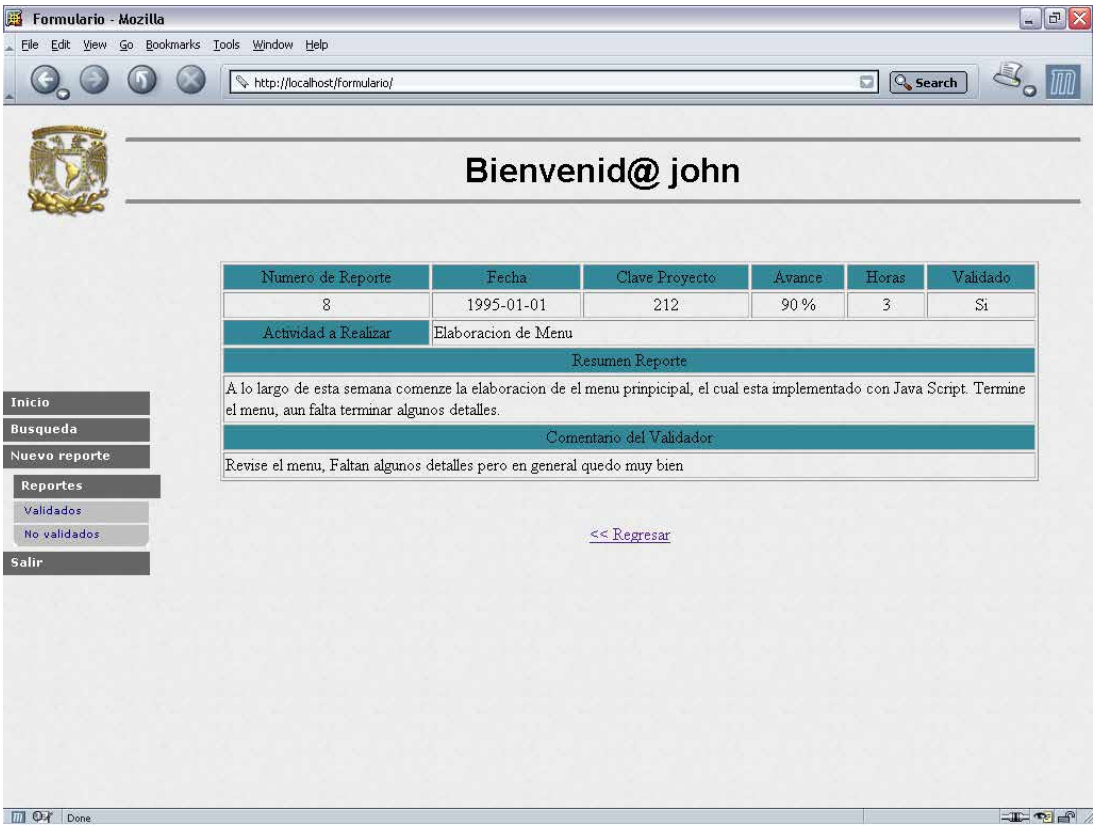


Figura 2.20 Resultado de la búsqueda de reporte validado numero 8

Figura 2.21 Resultado de Reportes No validados

Figura 2.22 Resultado de presionar sobre el enlace del reporte número 48

2.7.3 MÓDULO VALIDAR REPORTE

Con este módulo se llevará a cabo el control de información.

Validar Reporte consiste en que un usuario de mayor jerarquía, revise, valide y lleve un control de la información de todas aquellas actividades que ejecute otro usuario de menor jerarquía que el primero. Su página de inicio se muestra en la figura 2.23.

Este módulo realiza las tareas de validar los reportes de un usuario, administrar los registros de ese usuario pero principalmente llevar un control de aquellas actividades que han sido encomendadas al usuario de menor jerarquía.

La entrada al módulo ya se explico con anterioridad, por lo que explicaré otros puntos importantes,

Validar Reporte esta constituido por tres secciones:

- Búsqueda de reportes. Búsquedas de reportes que están almacenados en la base de datos, dicha búsqueda de reportes se pueden realizar de tres diferentes formas:
  - Por Fecha.
  - Por Número.
  - Por Clave de proyecto.
- Validar Reporte. Publica el último reporte de acuerdo a la fecha que ha sido introducido en la base de daros para ser validado.
- Reportes. Se exhiben todos los reportes almacenados en la base de datos, con la clasificación.
  - Validados.
  - No Validados.

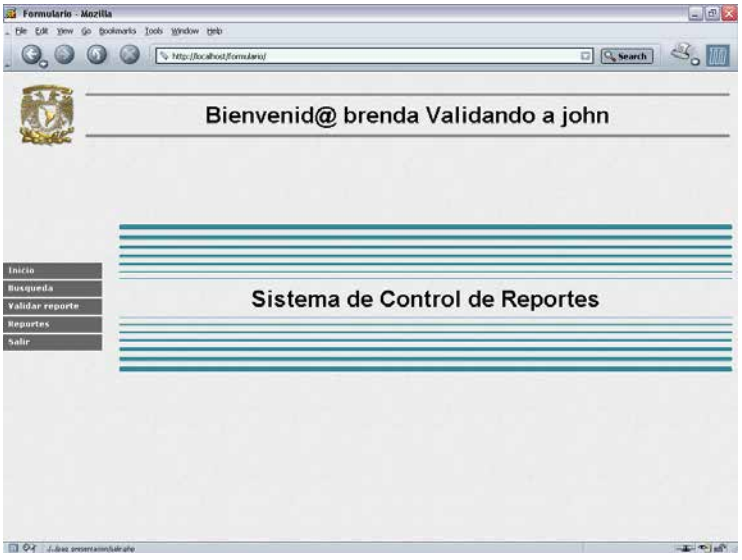


Figura 2.23 Página de Inicio del módulo Validar Reporte

2.7.3.1 BÚSQUEDA DE REPORTES

POR FECHA

La búsqueda de reportes *Por Fecha*, radica en seleccionar de dos listas el mes y el año del reporte que se requiere (Figura 2.24). El Resultado de la búsqueda publicará todos los reportes que se recuperen de la fecha seleccionada en forma de una tabla (Figura 2.25).

Para poder consultar un reporte en específico de la tabla mostrada en la Figura 2.25 basta con presionar sobre el número del reporte, para que se despliegue en toda la pantalla dicho reporte, y así poder consultar la información (Figura 2.26).

POR NÚMERO

La búsqueda por número se efectúa, introduciendo en el campo vacío el número de reporte que desea estudiar (ver Figura 2.27), al presionar sobre el botón *Buscar* se extiende el resultado en la pantalla (Figura 2.28).

POR CLAVE

La búsqueda por clave se efectúa, introduciendo en el campo vació la clave del proyecto de los reportes que desea consultar (ver Figura 2.29), basta con presionar sobre el botón *Buscar* para que enseguida se despliegue el resultado (Figura 2.30).

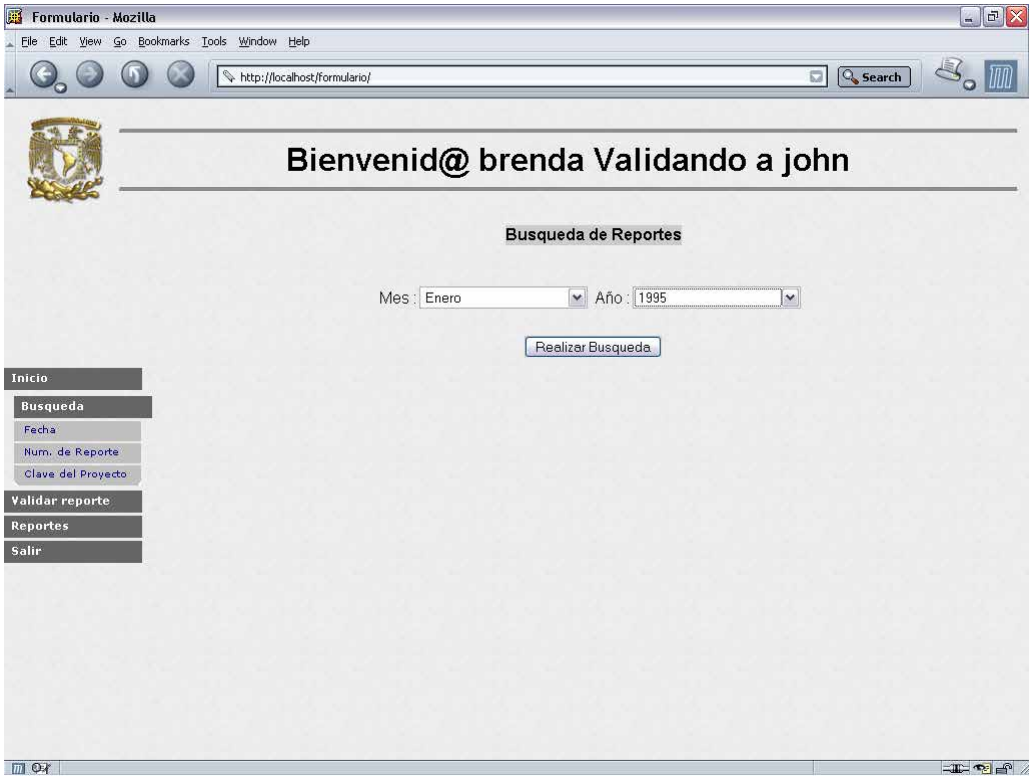


Figura 2.24 Búsqueda de reportes Por Fecha.

Figura 2.25 Resultado de la búsqueda

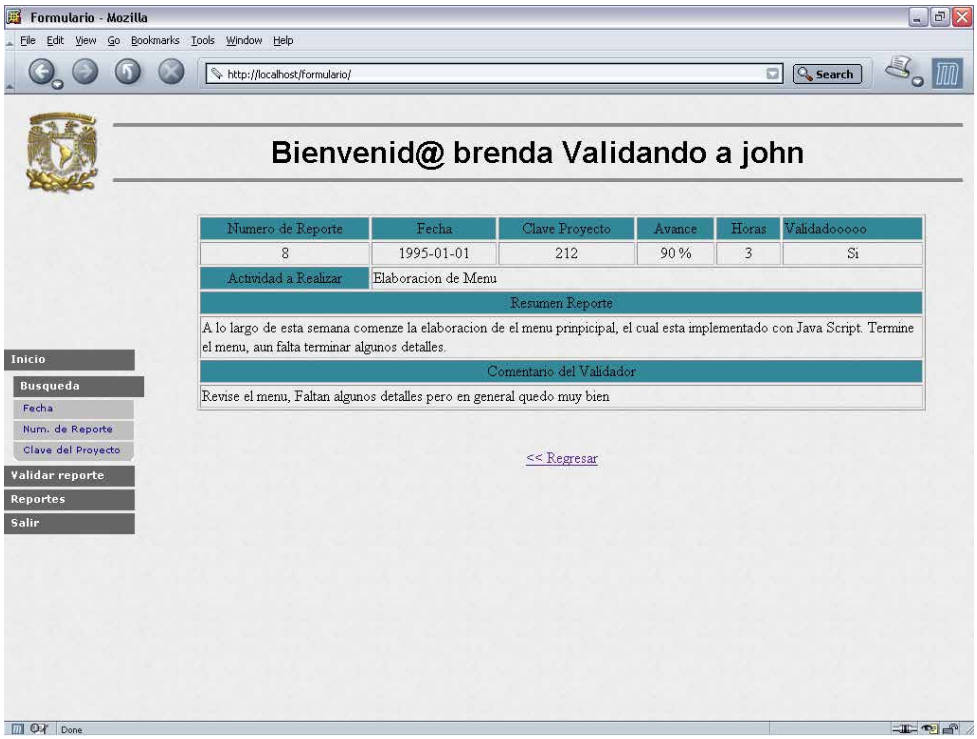


Figura 2.26 Información completa del reporte.

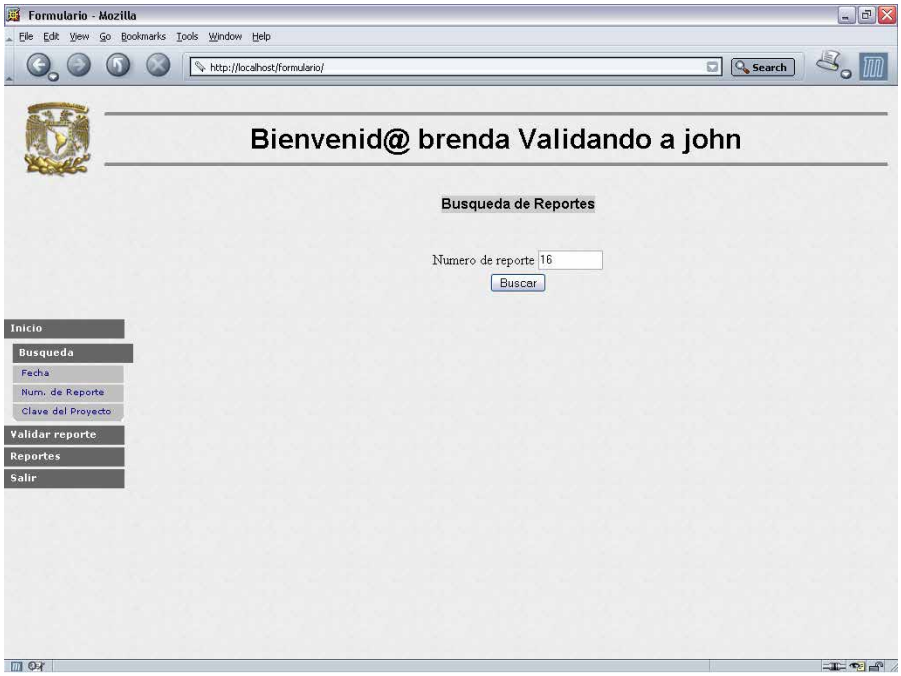


Figura 2.27 Búsqueda de reportes por número

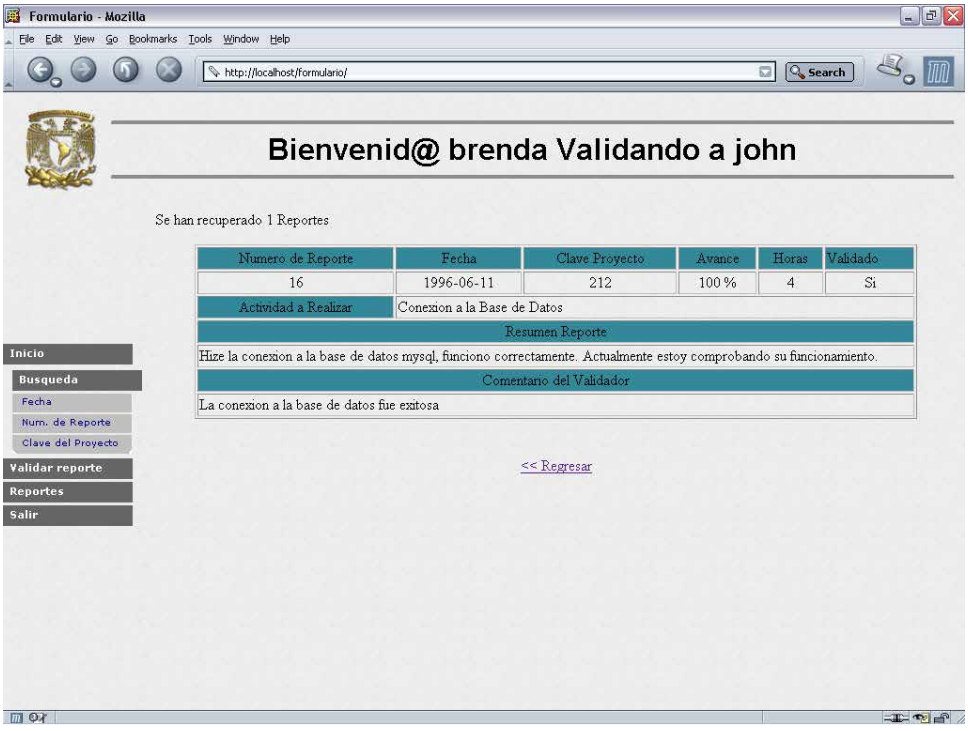


Figura 2.28 Resultado de la búsqueda por número.

Figura 2.29 Búsqueda de reportes por clave

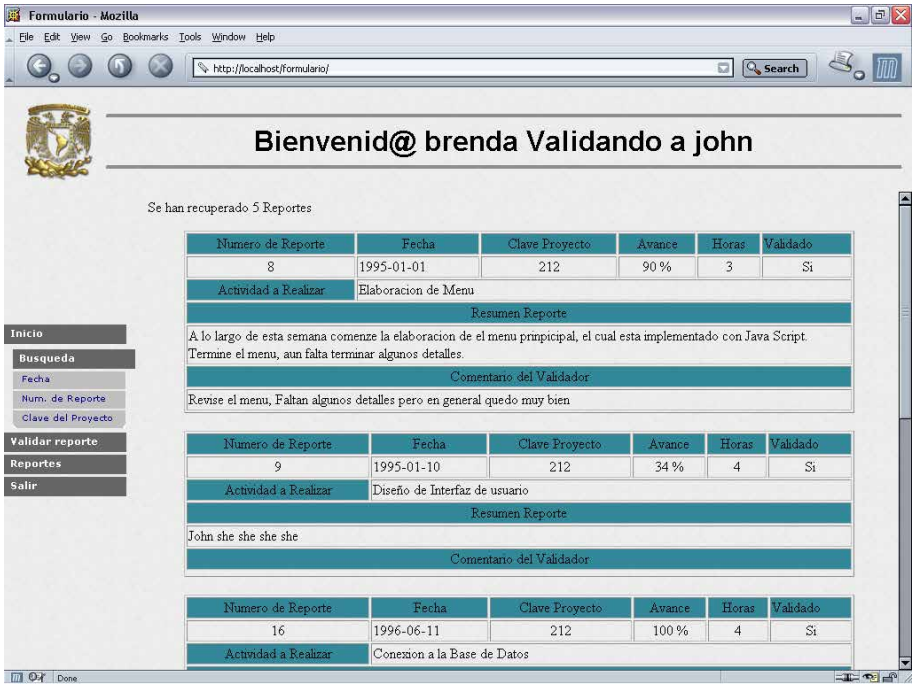


Figura 2.30 Resultado de la búsqueda de reportes por clave

2.7.3.2 VALIDAR REPORTE

Al presiona sobre *Validar Último Reporte* el resultado que se obtiene es el último reporte redactado por el usuario de acuerdo a su fecha de elaboración con el valor NO en el atributo Validado (Figura 2.31).

Validar Reporte consiste en aprobar o no aprobar las actividades que realizó a lo largo de cierto periodo otro usuario. También se puede agregar un comentario. En caso de No validar un reporte, este estará en la clasificación de reportes no validados, no significa que no se ha revisado, simplemente el validador no esta de acuerdo con las actividades descritas o por alguna otra razón personal.

El campo agregar comentario es opcional, es decir puede quedar en blanco y el sistema no marcara algún error.

Después de validar y agregar el comentario se debe dar clic sobre el botón aceptar para concluir la validación del reporte.

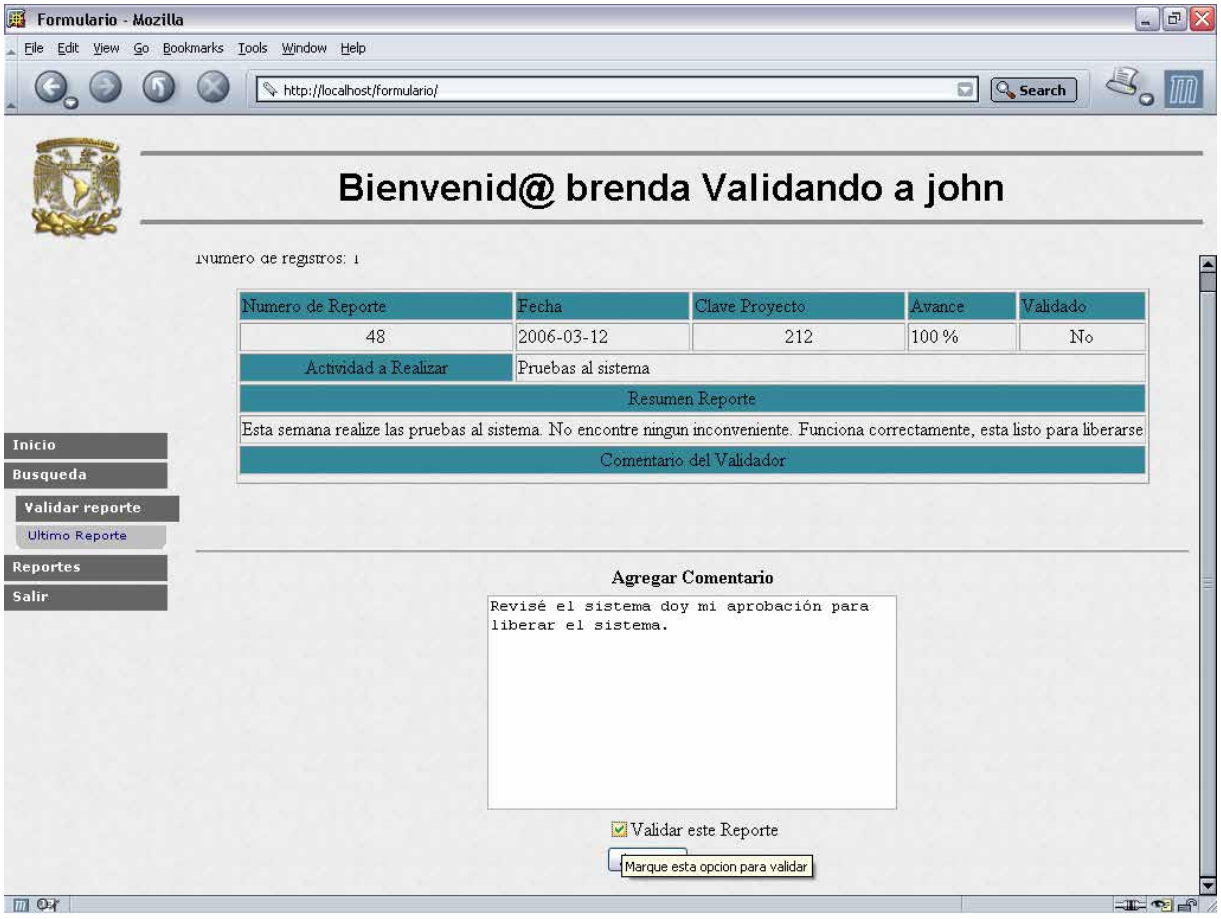


Figura 2.31 Reporte No validado

2.7.3.3 REPORTES

La funcionalidad de esta sección es la de mostrar todos los reportes del usuario que se esta validando clasificándolos en dos formas:

- Validados.
- No validados.

VALIDADOS

Con sólo dar clic sobre *Validados* mostrará una lista de todos aquellos reportes que ya han sido revisados y validados. La lista se va mostrar en forma de tabla con los siguientes atributos (Figura 2.32):

- Número de reporte.
- Fecha.
- Clave proyecto.
- Validado.

Para entrar a un reporte en específico se debe pulsar sobre el enlace. La Figura 2.33 visualiza el resultado de presionar el enlace del reporte número 48.

NO VALIDADOS

No validados es muy afín a *Reportes Validados*, la diferencia radica en que esta sección como su nombre lo indica descubre únicamente aquellos reportes que aún no han sido revisados o ya fueron revisados pero no validados.

Para poder validar un reporte se puede hacer de dos formas una de ellas ya se reviso en el punto 2.7.3.2, y la segunda es por esta sección, por lo cual podemos decir que *No Validados* tiene una doble funcionalidad:

1.- Listar aquellos reportes no validados.
2.- Validar reporte.

Para poder validar se de debe presiona sobre reportes *No validados* en seguida se va a desplegar en una tabla todos los reportes no validados (Figura 2.34), dando un clic sobre el enlace del reporte que se desea aprobar se desplegara otra pantalla, desde ésta se puede aprobar o desaprobar el reporte, con la opción de agregarle un comentario en caso de que así lo desee (Figura 2.35).

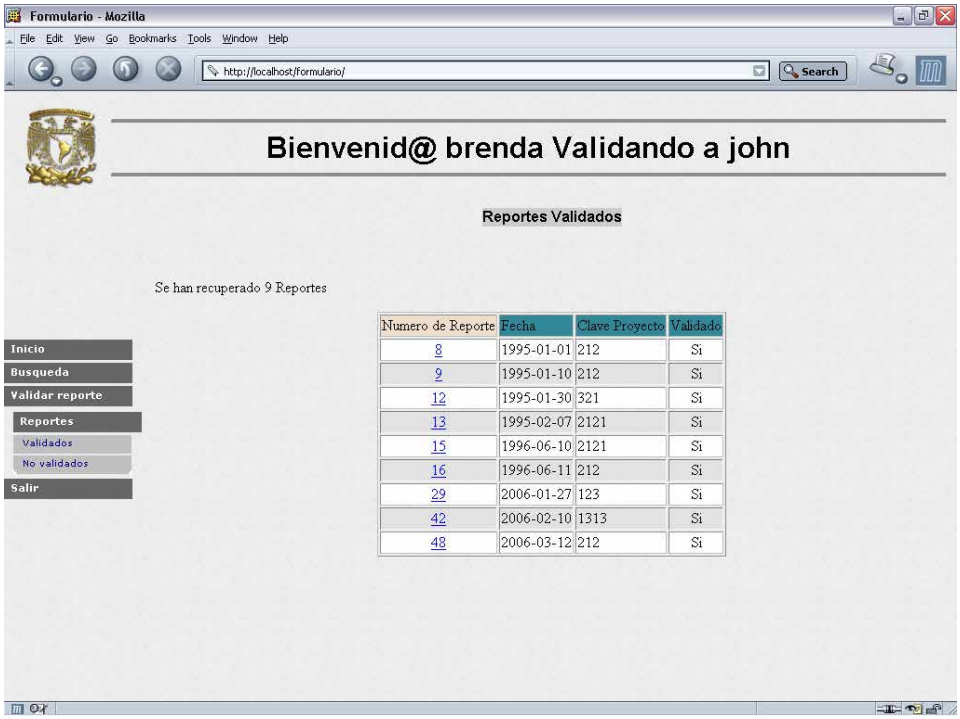


Figura 2.32 Lista de reporte validados

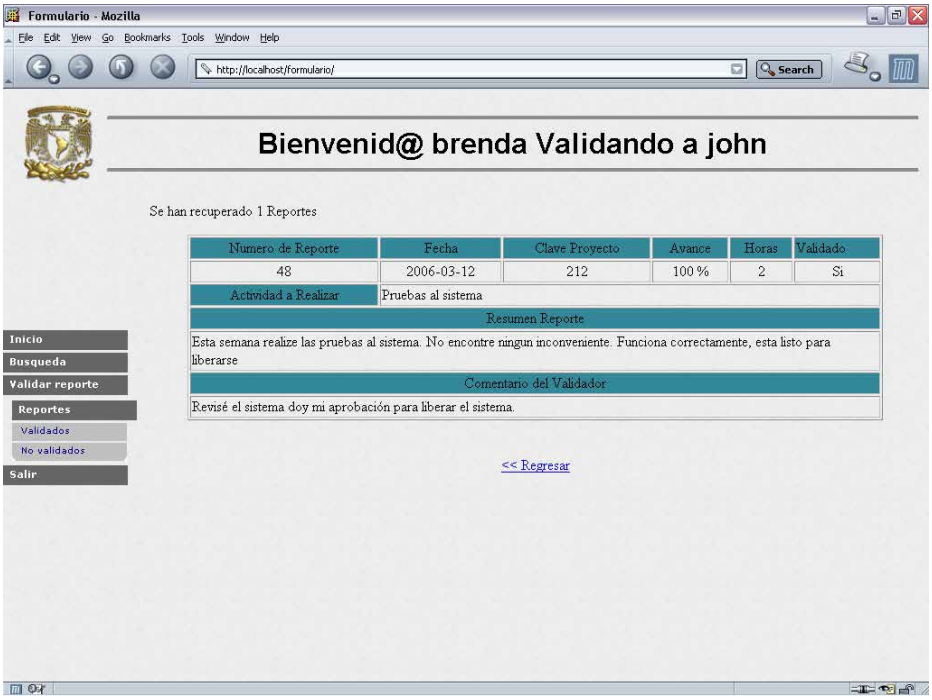


Figura 2.33 Resumen de reporte validado

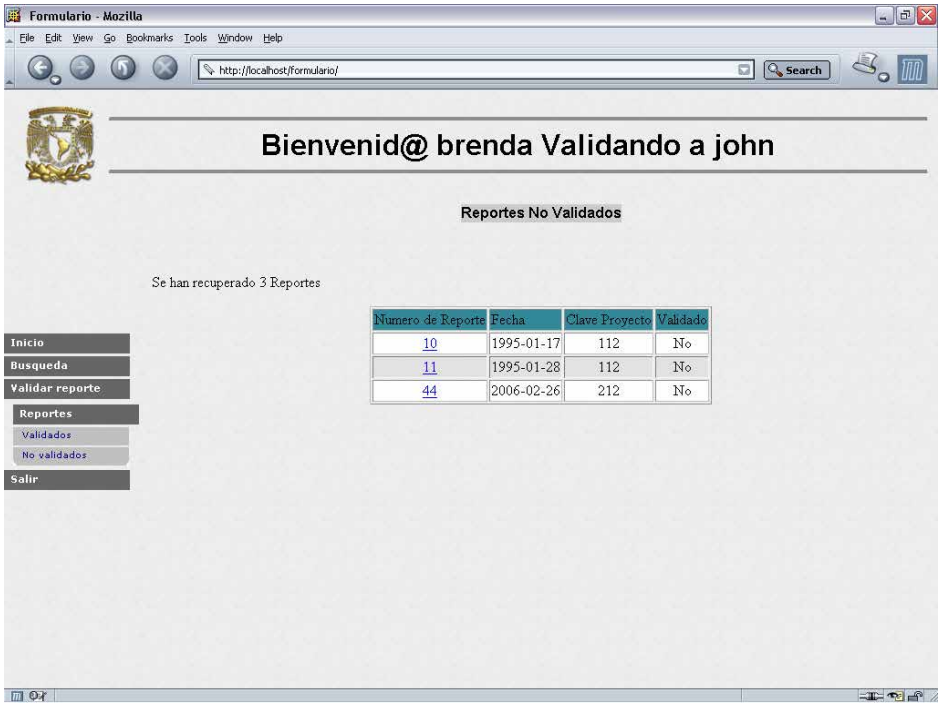


Figura 2.34 Lista de reportes No validados

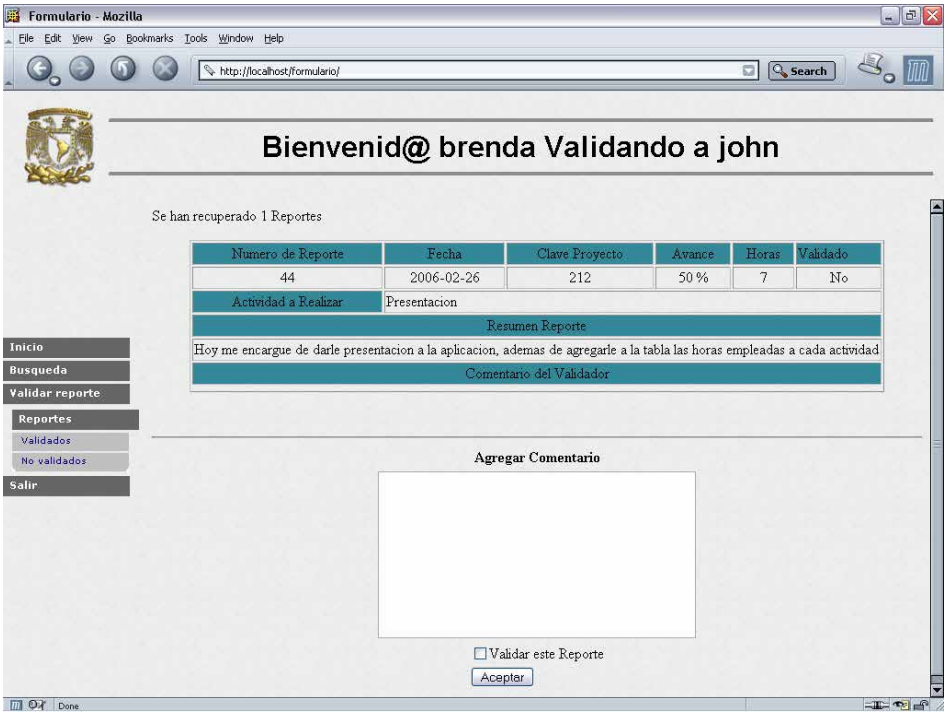


Figura 2.35 Resumen de reporte No validado

CONCLUSIÓN

La mayoría de las empresas que deciden adoptar tecnología de software libre se debe principalmente para beneficiar su economía. Después de analizar sus necesidades tecnológicas determinan si la tecnología de software libre puede cubrir sus necesidades, de esta forma, un porcentaje muy alto de empresas han resuelto por el software libre, siendo este una herramienta muy poderosa para la creación de sitios web, las cuales podemos adaptar a nuestras necesidades. El software libre cuenta con miles de usuarios alrededor del mundo mejorándolo de manera gratuita y lo mejor de todo sin tener que pagar costosas licencias.

El diplomado da las herramientas para poder construir y mantener soluciones requeridas para sitios de Internet, obteniendo una visión simple.

En la actualidad, PHP es una de las tecnologías más utilizadas en el mundo, y dos de sus grandes ventajas son que es gratuito y que puede emplearse en la mayoría de las plataformas, la mayoría de las veces PHP se utiliza para el desarrollo de sitios web, pero el uso que se le de dependerá en gran parte de las necesidades que posea el programador. MySQL, por su parte, es un servidor de bases de datos robusto, rápido y confiable, que, integrando a PHP, produce una combinación muy poderosa para el desarrollo de sitios web

Gracias al diplomado me he involucrado en los temas de desarrollo de sistemas para la Web con software libre. Me fueron de gran ayuda para emprender un camino lleno de posibilidades en la cual la capacitación (junto con el talento personal) ocupa un lugar fundamental y decisivo en mi vida profesional.

Del diplomado recibí una introducción y afianzamiento de diversas utilidades prácticas que pueden lograrse a partir de las herramientas de desarrollo por ejemplo el lenguaje de programación PHP y el servidor de bases de datos MySQL. Sin embargo todo este aprendizaje lo logré gracias a los estudios previos que hice en la Facultad de Estudios Superiores Aragón pues ahí fue donde adquirí los fundamentos de programación, diseño de bases de datos, desarrollo de sistemas de información, aprendí a planear, diseñar, organizar, sistemas para el procesamiento de datos y control de información.

Como futura ingeniera en computación debo de saber responder a las diversas necesidades y exigencias que se presentan en el campo laboral. Considero que el diplomado fue un complemento para poder lograr este objetivo.

GLOSARIO

Alias: Orden interna de bash. Permite sustituir una cadena por una sola palabra. Esto permite poner el nombre que queramos a un comando junto con sus parámetros. Ahorra tiempo en el trabajo diario, por lo que su uso es altamente recomendado. Puede ser algo tan sencillo como: alias rm='rm -i'. Por lo que cada vez que se teclee rm el shell lo sustituiría por rm -i.

Apache: Servidor de páginas web. Hoy por hoy líder del mercado de servidores, por delante de soluciones propietarias

AT&T: Compañía Estadounidense de telecomunicaciones. Una división de esta compañía, la Bells Lab, creó el primer Unix.

Background: Segundo plano. Se habla de proceso en segundo plano cuando se ejecuta sin la operabilidad del usuario o se pasa a modo suspendido.

Bash: Bourne Again Shell) Intérprete de comandos. Es el shell por defecto en la mayoría de las distribuciones de GNU/Linux de hoy en día. Se encarga de interpretar las ordenes que le demos para su proceso por el kernel.

Bind: Berkeley Internet Name Domain. Servidor de nombres de dominio.

Boot: Proceso de arranque en un sistema informático.

Case sensitivity: GNU/Linux distingue entre minúsculas y mayúsculas, por lo que se debe tener cuidado a la hora de teclear ordenes o nombres de ficheros.

Compilar: Proceso por el cual se "traduce" un programa escrito en un lenguaje de programación a lo que realmente entiende el ordenador.

Consola: Una consola la forman el teclado y el monitor del equipo donde se tiene instalado GNU/Linux . No confundir con terminal.

Cuenta: Una cuenta en un sistema Unix/Linux puede ser algo así como la llave de un taller comunitario. Es decir, se tiene una llave personal que permite acceder a ese taller y utilizar algunas de las herramientas del mismo. Donde además se tiene que atener a las normas que rijan en ese taller.

Demonio: Un programa que permanece en segundo plano ejecutándose continuamente para dar algún tipo de servicio. Ejemplos de demonio, son los servidores de correo, impresora, sistemas de conexión con redes etc.

Display: Variable de entorno, cuyo valor apunta al servidor Xwindow del usuario que lo esta ejecutando.

Dosemu: Emulador del sistema operativo DOS de Microsoft. Ejecuta gran parte de programas para este sistema operativo, incluidos juegos.

Emacs: Editor de texto. Aunque es su principal función, Emacs es hoy en día un programa muy extenso y con muchas utilidades, gracias a su soporte de plug-ins en lenguaje LISP.

Enlaces: Los enlaces o links permiten tener "copias" de un mismo archivo, ocupando solo el espacio del archivo real. Es decir, el enlace no es mas que otro archivo que apunta a el original.

Entrada/salida estándar: Por defecto la entrada de datos estándar se establece en el teclado y la salida de datos estándar en la pantalla del monitor, esto lo podemos variar a través de tuberías o redirecciones.

Ext2fs: Sistema de ficheros utilizado en GNU/Linux . Permite el uso de permisos para los ficheros y directorios, y tiende a fragmentarse mucho menos que los de otros sistemas operativos.

Filtro: Un filtro es un programa o conjunto de estos, que procesan una serie de datos generando una salida modificada conforme a lo que nosotros le especifiquemos.

Fdisk: Es un utilitario usado para crear, remover o modificar particiones en un disco duro.

FSF: Free Software Foundation. Fundación que pretende el desarrollo de un sistema operativo libre tipo UNIX. Fundada por Richard Stallman, empezó creando las herramientas necesarias para su propósito, de modo que no tuviera que depender de ninguna compañía comercial. Después vino la creación del núcleo, que todavía se encuentra en desarrollo.

Ftp: File Transfer Protocol. Servicio de Internet que permite el envío y la recepción de ficheros. Para su uso se necesita disponer de una cuenta en la maquina que va a recibir o enviar los ficheros.

gcc GNU C Compiler: El compilador estándar de la FSF.

Ghostcript: Programa encargado de la visualización de ficheros de texto con formato postcript.

GNOME: GNU Network Object Model Environment. Entorno de escritorio basado en las librerías GTK diseñadas para el programa de retoque fotográfico GIMP. Ofrece un entorno amigable y la posibilidad de que las aplicaciones intercambien datos entre si.

GNU: Gnu is Not Unix. Proyecto de la FSF para crear un sistema UNIX libre.

GNU/Linux: Sistema operativo compuesto de las herramientas GNU de la FSF y el núcleo desarrollado por Linus Torvalds y sus colaboradores.

GPL: General Public License. Una de las mejores aportaciones de la FSF. Es una licencia que protege la creación y distribución de software libre.

Groff: Versión GNU del programa nroff para el formateo de textos. Las páginas del manual en línea han sido escritas con este formato.

Host: Nombre de un ordenador en una red.

HTTP: HyperText Transfer Protocol. Protocolo de red para la transferencia de paginas de hipertexto, o lo que es lo mismo, paginas web como esta.

Init: Init es el primer proceso que se ejecuta en un sistema UNIX/Linux y el que inicia todos los procesos getty. Tiene varios estados, llamados niveles de ejecución, que determinan los servicios que pueden ofrecer.

Inode: Todos los archivos en UNIX/Linux tienen un inode que mantienen información referente al mismo, tal como situación, derechos de acceso, tamaño o tipo de fichero.

KDE: K Desktop Environment. Entorno de escritorio que integra gestor de ventanas propio y una barra de tareas y que al igual que GNOME permite la interacción entre sus aplicaciones.

Kernel: Núcleo. Parte fundamental de un programa, por lo general de un sistema operativo, que reside en memoria todo el tiempo y que provee los servicios básicos. Es la parte del sistema operativo que está más cerca de la máquina y puede activar el hardware directamente o unirse a otra capa de software que maneja el hardware.

Lilo: LInux LOader. Programa que permite elegir que sistema operativo arrancar, en el caso de tener varios.

Linux: Núcleo del sistema operativo GNU/Linux

Login: Programa encargado de la validación de un usuario a la entrada al sistema. Primero pide el nombre del usuario y después comprueba que el password sea el asignado a este.

Man: Manual en línea del sistema. Aquí se puede buscar casi cualquier cosa relacionada con el sistema, sus comandos, las funciones de biblioteca, etc.

Mbr: Master Boot Record. Tabla de información referente al tamaño de las particiones.

Monousuario: Sistema informático que solo admite el trabajo con una persona.

Montar: Poner un dispositivo o un sistema de ficheros en disposición de ser usado por el sistema.

Multitarea: Capacidad de un sistema para el trabajo con varias aplicaciones al mismo tiempo.

Multiusuario: Capacidad de algunos sistemas para ofrecer sus recursos a diversos usuarios conectados a través de terminales.

Núcleo: Parte principal de un sistema operativo, encargado del manejo de los dispositivos, la gestión de la memoria, del acceso a disco y en general de casi todas las operaciones del sistema que permanecen invisibles para nosotros.

Password: Palabra clave personal, que permite el acceso al sistema una vez autentificada con la que posee el sistema en el fichero passwd.

Path: Variable del entorno, cuyo valor contiene los directorios donde el sistema buscara cuando intente encontrar un comando o aplicación. Viene definida en los ficheros .bashrc o .bash_profile de nuestro directorio home.

Permisos: Todos los archivos en UNIX/Linux tienen definido un set de permisos que permiten establecer los derechos de lectura, escritura o ejecución para el dueño del archivo, el grupo al que pertenece y los demás usuarios.

Proceso: Programa en ejecución en un sistema informático.

Prompt: El prompt es lo siguiente que se ve al entrar al sistema, una línea desde donde el sistema indica que esta listo para recibir órdenes, que puede ser tan sencilla como: $

Root: Persona o personas encargadas de la administración del sistema Tiene TODO el privilegio para hacer y deshacer, por lo que su uso para tareas que no sean absolutamente necesarias es muy peligroso. Otra definición sería la de que es el directorio inicial de un sistema de archivo

Señales: Las señales son eventos que se hacen llegar a un proceso en ejecución para su tratamiento por este. Las señales las pueden enviar los usuarios u otros programas a otros programas. Tienen diferentes valores, y en función a esos valores el proceso que las recibe actúa de una manera u otra.

Shell: Traducido del inglés concha o caparazón. El shell es el intérprete de comandos que se establece entre nosotros y el kernel. Hay muchos tipos de shell cada uno con sus propias características, sin embargo el estándar en GNU/Linux es el shell bash ya que es el que forma parte del proyecto GNU.

Superusuario: Persona o personas encargadas de la administración del sistema Tiene TODO el privilegio para hacer y deshacer, por lo que su uso para tareas que no sean absolutamente necesarias es muy peligroso. Otra definición sería la de que es el directorio inicial de un sistema de archivo

Swap: Memoria virtual. Espacio de disco duro que utiliza el kernel en caso de necesitar mas memoria de la que tengamos instalada en nuestro ordenador

Telnet: Servicio que permite la conexión a otra computadora de la red, pasando nuestro sistema a ser una terminal de ese ordenador.

Terminal: Una terminal es un teclado y una pantalla conectados por cable u otro medio a un sistema UNIX/Linux, haciendo uso de los recursos del sistema conectado.

Tubería: Las tuberías son como conexiones entre procesos. La salida de un proceso la encadenamos con la entrada de otro, con lo que se puede procesar unos datos en una sola línea de comando.

Unix: Sistema operativo creado por AT&T a mediados de los 70

Vi: Editor de texto muy potente aunque algo complejo al principio. Es el editor por defecto en casi todas las distribuciones.

BIBLIOGRAFÍA

BEN, Laurie y LAURIE, Peter, *Apache: The Definitive Guide*, Ed. O'Reilly, 2002, 588 pp

BROTEN, Rich y COAR Ken, *Servidor Apache*, Ed. Prentice Hall, España, 2000, 610 pp

CABEZAS G, Luis Miguel, *PHP 5*, Ed. Anaya, 2004, 384 pp.

CHARTE O, Francisco, *PHP 5*, Ed. Anaya , España, 2004, 352 pp

COOPER B, Richard et al, *Apache Administrator's Handbook*, Ed. Sams Publishing , 2002, 422 pp

GARCÍA S, Ji, *HTML4*, Ed. Osborne Mc Graw Hill, España, 2001, 452 pp

GUTIÉRREZ G, Bravo, *PHP5 a través de ejemplos*, Ed. Alfaomega, México D.F., 2005, 552 pp

EDWARD U, Larry, *Php and Mysql for Dynamic Web Sites*, Ed. Peachpit Press, 2003, 400 pp

MINERA, Francisco José, *PHP y MySQL*, Ed. Users.code, Buenos Aires Argentina, 2005, 424pp

PETERSAN, Richard, *Manual de referencia Linux*, Ed. Osborne Mc Graw Hill, España, 2000, 1306 pp

Pagina web de php http://www.php.net/

Pagina web de apache http://www.apache.org/

Pagina web de mysql http://www.mysql.com/

Pagina web de netcraf http://news.netcraft.com/

Pagina web oficial de la Free Software Foundation. http://www.gnu.org/

Sitio oficial de Slackware http://www.slackware.com/

Tutorial HTML http://www.htmlcodetutorial.com/

Tutorial php http://www.programacion.com/php/