UNIVESIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES "ARAGÓN"

Ingeniería en computación

"Desarrollo de Sistemas en Línea con Software Libre en Linux"

TRABAJO ESCRITO

EN LA MODALIDAD DE SEMINARIOS Y CAPACITACIÓN PROFESIONAL PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

EDGAR IGNACIO PEÑA FLORES

ASESOR: ING. CÉSAR GERMÁN ROSAS

México, 2006

Agradecimiento

A mis padres:

Por su inmenso cariño, comprensión,
apoyo y por creer en mí.

Índice

Introducción 8
1. Sistema operativo Linux 10
1.1 Características de Linux 10
1.2 Algunas distribuciones 11
1.3 Componentes del sistema operativo Linux 11
1.4 El sistema de archivos 12
1.5 Estructura de directorios en Linux 12
1.6 Comandos básicos 13
1.6.1 Reconocimiento del servidor y del sistema 13
1.6.1.1 hostname 13
1.6.1.2 uname 14
1.6.2 Creación y gestión de archivos y directorios 14
1.6.2.1 ls 14
1.6.2.2 cd 15
1.6.2.3 pwd 15
1.6.2.4 touch 15
1.6.2.5 mkdir 15
1.6.2.6 cp 15
1.6.2.7 rm 16
1.6.2.8 mv 16
1.6.3 Trabajo con archivos 16
1.6.3.1 cat 16
1.6.3.2 more 17
1.6.3.3 head 17
1.6.3.4 tail 17
1.6.3.5 sort 17
1.6.3.6 wc 18
1.7 Rutas absolutas y relativas 18
1.8 Atributos de los archivos 19
1.8.1 Algunos tipos de archivos 19
1.8.2 Permisos 19
1.8.2.1 chmod 20
1.9 Redireccionamiento 21
1.9.1 Redireccionamiento de salida 21
1.9.2 Redireccionamiento de entrada 21
1.9.3 Redireccionamiento de un programa a otro 21
1.9.4 Redireccionamiento no destructivo o de adición 22
1.9.5 Redireccionamiento condicionado 22
1.10 Ligas 22
1.10.1 Liga suave 22
1.10.2 Liga dura 23
2. Instalación y administración de Linux 24
2.1 Importancia de la administración 24
2.2 Tareas administrativas comunes 24
2.3 Actividades del administrador 24
2.4 Conocimientos de un administrador 24

2.5 Instalación de Linux 25
2.5.1 Proceso de instalación de Slackware 9.0 27
2.6 Dar de alta o baja el sistema 30
2.6.1 shutdown 30
2.7 Cuentas de usuario 31
2.8 Agregando usuarios 32
2.8.1 useradd 32
2.9 Borrando usuarios 32
2.9.1 userdel 32
2.10 Grupos 33
2.10.1 groupadd 33
2.11 Dispositivos y Linux 33
2.11.1 Disquetes 34
2.11.2 Unidades de CD-ROM y DVD 34
3. Programación en Shell 35
3.1 Primer programa de shell 35
3.2 Paso de parámetros a un programa de shell 35
3.3 Algunas variables especiales del shell 36
3.4 shift 37
3.5 read 38
3.6 Operadores aritméticos 38
3.7 Operadores relacionales 39
3.8 Operadores lógicos 39
3.9 expr 39
3.10 test 40
3.10.1 Evaluación de archivos con *test* 40
3.10.2 Evaluación de cadenas 41
3.10.3 Evaluaciones numéricas 41
3.11 if 42
3.12 case 42
3.13 while 43
3.14 until 44
3.15 for 45
3.16 Funciones 46
4. Introducción a HTML 48
4.1 Sintaxis del HTML 48
4.2 Partes de un documento HTML 48
4.3 Formato de párrafos en HTML 49
4.4 Formato de texto 49
4.4.1 Negritas 49
4.4.2 Itálicas 49
4.4.3 Subrayado 49
4.4.4 Subíndices y supraíndices 50
4.5 Color, tamaño y tipo de letra 50
4.6 Atributos para páginas 50
4.7 Listas 51
4.7.1 Listas desordenadas 51
4.7.2 Listas ordenadas 52
4.7.2 Listas de definición 53

4.8 Enlaces 53
4.8.1 Enlaces internos 54
4.8.2 Enlaces locales 54
4.8.3 Enlaces remotos 54
4.8.4 Enlaces a direcciones de correo 55
4.9 Imágenes en HTML 55
4.10 Tablas en HTML 55
4.11 Formularios 57
4.11.1 Cajas de texto 58
4.11.2 Área de texto 58
4.11.3 Lista de opciones 59
4.11.4 Botones de radio 60
4.11.5 Cajas de validación 60
4.11.6 Botón de envió 61
4.11.7 Botón de borrado 61
4.11.8 Datos ocultos 61
5. Programación con PHP 62
5.1 ¿Qué es un servidor WWW? 62
5.2 ¿Cómo funciona HTTP? 62
5.3 Servidores WWW Populares 62
5.4 Criterios de Selección 63
5.5 Características de Apache 63
5.6 Instalación de Apache 63
5.7 Configuración del Servidor 64
5.7.1 User 64
5.7.2 Group 64
5.7.3 Port 64
5.7.4 ServerAdmin 64
5.7.5 DocumentRoot 65
5.7.6 ServerRoot 65
5.7.7 MinSpareServers 65
5.7.8 MaxSpareServers 65
5.7.9 StartServers 65
5.7.10 MaxClients 66
5.7.11 ErrorDocument 66
5.7.12 PidFile 66
5.7.13 HostnameLookups 66
5.7.14 <Directory> 67
5.7.15 Options 67
5.8 CGI (Common Gateway Interface) 68
5.8.1 ScriptAlias 68
5.8.2 AddHandler 68
5.9 Manejo de sitios virtuales 69
5.9.1 <VirtualHost> 69
5.10 Introducción a PHP 70
5.11 Tareas principales de PHP 70
5.12 Sintaxis de PHP 70
5.13 Funcionamiento de una página PHP 70
5.14 Variables en PHP 71

5.14.1 Variables numéricas 71
5.14.2 Variables alfanuméricas 71
5.14.3 Arreglos (arrays) 71
5.15 Variables de sistema en PHP 71
5.16 Variables superglobales 72
5.17 Manejo de arreglos con PHP 73
5.18 Funciones básicas para el manejo de arreglos 74
5.19 Cadenas 75
5.19.1 Concatenación de cadenas 76
5.19.2 Funciones básicas para el manejo de cadenas 76
5.20 Funciones 77
5.21 Operadores 78
5.21.1 Operadores aritméticos 78
5.21.2 Operadores de comparación 78
5.21.3 Operadores lógicos 78
5.21.4 Operadores de incremento 79
5.21.5 Operadores combinados 79
5.22 if-else 79
5.23 while 80
5.24 for 80
5.25 foreach 81
5.26 Paso de variables por la URL 82
5.27 Procesar variables de Formularios 83
5.28 Sesiones 84
5.28.1 Funciones importantes para la gestión de sesiones 85
6. PHP con MYSQL 86
6.1 Instalación de MYSQL 4.0.26 86
6.2 Introducción a SQL 87
6.2.1 Tipos de datos comunes 87
6.2.2 Insertar un nuevo registro 87
6.2.3 Eliminar un registro 87
6.2.4 Actualizar un registro 88
6.2.5 Crear una tabla 88
6.2.6 Seleccionar algunos campos 89
6.2.7 Seleccionar todos los campos de la tabla 89
6.2.8 Distinct 89
6.2.9 Order by 90
6.2.10 Operadores 90
6.3 Tipos de datos en Mysql 91
6.4 Comandos básicos de Mysql 92
7. Introducción a la seguridad en cómputo 93
7.1 Algunos conceptos de seguridad 93
7.2 Estructura de la seguridad informática 94
7.3 Control de Acceso 95
7.4 Administración básica de la seguridad 96
7.5 Monitoreo de Sistemas 96
7.6 Procesos 97
7.7 Seguridad en redes 98
7.8 Detección de intrusos. 98

7.9 Firewalls 99
8. PHP con PostgreSQL 101
8.1 Instalación de PostgreSQL 101
8.2 Tipos de datos 102
8.3 Administración de una base de datos en PostgreSQL 103
8.3.1 Acceso a la base de datos 103
8.3.2 Destrucción de una base de datos 103
8.3.3 Copia de seguridad y restauración 104
8.3.4 Bases de datos grandes 104
8.4 Construcción de bases de datos con PostgreSQL 105
8.4.1 Creación de tablas 105
8.4.2 Constraints 105
8.4.3 Insertando datos 107
8.4.4 Actualizando datos 107
8.5 Acceso a bases de datos PosgreSQL con PHP 108
9. Caso Práctico (Sistema de Administración de Clientes y Servicios) 110
9.1 Motivación del proyecto (Situación actual) 110
9.2 Declaración de la visión 110
9.3 Alcance 111
9.4 Análisis del Sistema de Administración de Clientes y Servicios (SACS) 111
9.5 Relación de requerimientos del área usuaria 112
9.6 Descripción de los actores 112
9.7 Diagrama de Casos de uso 113
9.7.1 Documento de casos de uso 114
9.8 Casos de uso relativos al cliente 115
9.8.1 Realizar registro 115
9.8.2 Registrar Autenticación 116
9.8.3 Cotizar productos 117
9.8.4 Consultar Pagos 123
9.8.5 Realizar pedido 124
9.8.*6* Consultar productos 125
9.9 Casos de uso relativos al administrador 126
9.9.1 Registrar autenticación 126
9.9.2 Administrar catálogos 127
9.9.3 Generar reportes 133
9.9.4 Administrar contrato de alojamiento 135
9.10 Casos de uso relativos al personal 136
9.10.1 Registrar Autenticación 136
9.10.2 Consultar Pedidos 137
9.10.3 Cotizar Productos 138
9.10.4 Realizar Pedido 139
9.10.5 Administrar catálogo de clientes 139
9.10.6 Administrar contrato de alojamiento 141
9.11 Diagramas de secuencia relativos al cliente 143
9.11.1 Realizar registro 143
9.11.2 Registrar Autenticación 144
9.11.3 Cotizar productos 145
9.11.4 Consultar pagos 147

9.11.5 Realizar pedido 148
9.11.6 Consultar productos 149
9.12 Diagramas de secuencia relativos al administrador 150
9.12.1 Registrar autenticación 150
9.12.2 Administrar catálogos 151
9.12.3 Generar reportes 153
9.12.4 Administrar contrato de alojamiento 153
9.13 Diagramas de secuencia relativos al personal 154
9.13.1 Registrar autenticación 154
9.13.2 Consultar pedidos 155
9.13.3 Cotizar Productos 155
9.13.4 Realizar pedido 156
9.13.5 Administrar catálogo de clientes 157
9.13.6 Administrar contrato de alojamiento 159
9.14 Diagrama Entidad Relación 160
9.15 Diccionario de datos 161
Conclusión 170
Bibliografía 171

Introducción

Ente los años 60 y 70's del siglo XX, el software no era considerado como un producto sino un añadido que los vendedores de las grandes computadoras de la época (los mainframes) aportaban a sus clientes para que pudieran usarlas.

En dicha cultura, era común que los programadores y desarrolladores de software compartieran libremente sus programas unos con otros. Muy similar a la forma de hacer ciencia, es decir, los científicos presentan publicaciones e investigaciones y otros científicos interesados las estudian, las refutan o simplemente confirman y si todo es correcto se dedican a hacer nuevas investigaciones basadas en las anteriores.

Pero a finales de los 70's, las compañías iniciaron el hábito de imponer restricciones a los usuarios, con el uso de acuerdos de licencia. El inconveniente que presenta este tipo de software es muy visible, el costo de las licencias muy elevado.

En 1984, Richard Stallman comenzó a trabajar en el proyecto *GNU*[*], y un año más tarde fundó la Free Software Foundation (FSF). Stallman introdujo una definición para *free software* y el concepto de *copyleft*[†], el cual se desarrolló para dar a los usuarios libertad y para restringir las posibilidades de apropiación del software.

De acuerdo con esa definición el software es "libre" si garantiza las siguientes libertades:

- "Libertad 0", ejecutar el programa con cualquier propósito (privado, educativo, público, comercial, etc.).
- "Libertad 1", estudiar y modificar el programa (para lo cuál es necesario poder acceder al código fuente).
- "Libertad 2", copiar el programa de manera que se pueda ayudar al vecino o a cualquiera.
- "Libertad 3", mejorar el programa, y hacer públicas las mejoras, de forma que se beneficie toda la comunidad.

Las libertades 1 y 3 se refieren a que se tenga acceso al código fuente. La libertad 2 hace referencia a la libertad de modificar y redistribuir el software libremente licenciado bajo algún tipo de licencia de software libre que beneficie a la comunidad.

---

[*] El **proyecto GNU** iniciado por Richard Stallman tiene el objetivo de crear un sistema operativo completo libre: El sistema GNU. GNU es un acrónimo recursivo que significa "GNU No es UNIX".

[†] **Copyleft** es un término que se asocia a una clase de licencias que, aplicadas a creaciones como el software y otras obras creativas, permite respecto de dichas obras tener las libertades de uso, copia, modificación y redistribución de lo modificado y, además, que cualquier obra derivada continúe manteniendo la licencia que permite tales libertades.

Actualmente existe una gran cantidad de software libre, cada vez mayor, disponible bajo licencias de software libre.

Algunos proyectos notables del software libre incluyen los kernel de los sistemas operativos *GNU-Linux* y *BSD*, los compiladores *GCC*, el depurador *GDB* y las bibliotecas de *C,* el servidor de nombres *BIND*, el servidor de correo, *Sendmail*, el Servidor Web *Apache*, los sistemas de bases de datos relacionales *MySQL* y *PostgreSQL*, los lenguajes de programación *Perl, Python, Tcl* y *PHP*, el sistema *X Window*, los entornos de escritorio *GNOME* y *KDE*, la suite de ofimática *OpenOffice*, el navegador *Mozilla*, el servidor de archivos *Samba*, y el editor de gráficos *GIMP*.

Las anteriores ejemplificaciones hacen referencia al software muy interesante y de gran calidad con el que se pueden desarrollar aplicaciones robustas, seguras.

El presente trabajo escrito, describe a grandes rasgos los módulos cursados en el diplomado “Desarrollo e implementación de sistemas con software libre en Linux”.

1. Sistema operativo Linux

En 1990, Linus Torvalds, un estudiante de 23 años de la Universidad de Helsinki, en Finlandia, comenzó a desarrollar un proyecto basado en *MINIX*[*]. Linus quería llevar acabo sobre una computadora con procesador Intel 80386, un sistema operativo tipo *UNIX* que ofreciera más capacidades que el limitado *MINIX*.

Este proyecto personal desembocó en octubre de 1991 con el anuncio de la primera versión de Linux capaz de ejecutar *BASH*[†] (Bourne Again Shell) y *GCC* (GNU Compiler Collection). Poco tiempo después en enero de 1992 se adoptó la *GPL* (Licencia Pública General) para Linux. Ésta añade libertades de uso a Linux, permitiendo su modificación, redistribución, copia y uso ilimitado.

Es importante mencionar que la historia de Linux está fuertemente ligada a la del Proyecto *GNU* que inició en 1993 y tiene como objetivo el desarrollo de un sistema UNIX completo, pero totalmente hecho con Software Libre.

Cuando la primera versión del núcleo Linux fue liberada, el proyecto GNU había producido varios de los componentes del sistema: un intérprete de comandos, una biblioteca C y un compilador. Pero el proyecto GNU aún no contaba con el núcleo que definiría un sistema operativo. Sin embargo, el núcleo creado por Torvalds, llenó el hueco final que el sistema operativo GNU exigía.

Subsecuentemente, miles de programadores voluntarios alrededor del mundo han participado en el proyecto, mejorándolo continuamente. Torvalds y otros desarrolladores de los primeros días de Linux adoptaron los componentes de GNU para trabajar con el núcleo de Linux, creando un sistema operativo completamente funcional.

1.1 Características de Linux

Las características más importantes del sistema operativo Linux son las siguientes:

- **Multiproceso.** Permite la ejecución de varias aplicaciones simultáneamente.
- **Multiusuario.** Diferentes usuarios pueden acceder a los recursos del sistema, de manera simultánea.
- **Multiplataforma.** Funciona con la mayoría de las plataformas del mercado: Intel 386/486, Motorola 680, Sun Sparc, etc.

---

[*] **MINIX** es un clon del sistema operativo UNIX, distribuido junto con su código fuente y desarrollado por el profesor Andrew S. Tanenbaum en 1987.

[†] **Bash** es un Intérprete de comandos de tipo Unix (*shell*) escrito para el proyecto GNU. Su nombre es un acrónimo de **Bourne-Again SHell**: un juego de palabras de Bourne shell *sh*, el cual fue el intérprete original de UNIX.

- **Shells programables.** Esto lo convierte en un sistema operativo flexible.

Además de estas características, la creciente popularidad de Linux entre otras razones se debe a su *estabilidad*, al *acceso de las fuentes*, a la *independencia del proveedor*, a la *seguridad*, a la rapidez con que incorpora los nuevos adelantos (por ejemplo IPv6[*], microprocesadores de 64 bits), a la *escalabilidad* (se pueden crear clusters de cientos de computadoras), a la activa comunidad de desarrollo que hay a su alrededor.

1.2 Algunas distribuciones

Una *distribución*[†] incluye todo lo necesario para instalar un sistema Linux bastante completo y en la mayoría de los casos trae un programa de instalación que nos facilitará esta tarea.

Actualmente hay una gran cantidad de distribuciones, pero las más comunes son las siguientes:

- Slackware
- Redhat
- Debian
- SuSE
- Caldera
- Mandrake
- Gentoo
- MkLinux
- Ubuntu
- Fedora Core

1.3 Componentes del sistema operativo Linux

Linux está compuesto por cuatro capas, la capa interna está conformada por el ***hardware*** (el conjunto de piezas físicas del equipo de cómputo).

La segunda capa es el ***kernel*** o el núcleo del sistema. Su función principal es interpretar las instrucciones proporcionadas por el usuario y convertirlas en lenguaje máquina e indicarle al hardware lo que tiene que hacer con dicha información.

La tercera capa está formada por el grupo de los ***shells*** o ***intérpretes de comandos***, los cuales funcionan como la interfaz entre el usuario y el kernel,

[*] **IPv6** es la versión 6 del Protocolo de Internet (Internet Protocol), un estándar de red encargado de dirigir y encaminar los paquetes a través de una red.

[†] Una **distribución** no es otra cosa que una recopilación de programas y archivos organizados y preparados para su instalación. Regularmente estas distribuciones se pueden obtener a través de Internet o se pueden adquirir los CD's de las mismas.

proporcionan las herramientas para que el usuario se pueda comunicar con el núcleo del sistema Linux.

Por ultimo la cuarta capa donde se encuentra ***el usuario*** junto con ***los programas y aplicaciones*** que se tengan en el sistema (ejemplo, hojas de cálculo, procesadores de texto, manejadores de bases de datos, etc.).

1.4 El sistema de archivos

Linux está compuesto de un *sistema de archivos*[*] jerárquico en el cual no existen unidades de disco como en el sistema operativo Windows, en su lugar cada unidad de almacenamiento así como cada dispositivo de hardware son reconocidos como un ***archivo*** o ***directorio*** dentro del sistema.

Existe sólo una raíz en el sistema la cual representa la parte más alta de la estructura de directorios y es conocida como ***root*** o raíz, es representada por el símbolo ***"/"*** a partir de este punto se desprenden diferentes ramas o directorios.

El sistema de archivos nativo de Linux es el *EXT2*, pero actualmente existen muchos sistemas de archivos que presentan más ventajas que *EXT2,* por ejemplo *EXT3, ReiserFS* o *XFS*.

1.5 Estructura de directorios en Linux

Es importante conocer de qué forma están organizados los directorios en Linux, de esta manera se podrá ubicar fácilmente los recursos que necesitemos en nuestro sistema. Aquí está una pequeña tabla que describe los principales directorios de Linux.

[*] Un **sistema de archivos** consta de tipos de datos abstractos, que son necesarios para el almacenamiento, organización jerárquica, manipulación, navegación, acceso y consulta de datos.

Directorio	Descripción
/	Es la raíz del sistema de directorios, aquí se monta la partición principal de Linux.
/etc	Contiene los archivos de configuración de la mayoría de los programas.
/home	Contiene los archivos personales de los usuarios.
/bin	Contiene los comandos básicos y muchos programas.
/dev	Contiene archivos simbólicos que representan partes del hardware, tales como discos duros, memoria, etc.
/mnt	Contiene subdirectorios donde se montan otras particiones de disco duro, CD-ROM, etc.
/tmp	Contiene archivos temporales o de recursos de programas.
/usr	Programas y librerías instalados con la distribución.
/usr/local	Programas y librerías instalados por el administrador.
/sbin	Comandos administrativos.
/lib	Librerías y módulos de kernel
/var	Archivos de log (registros de actividad) de programas, bases de datos, contenidos del servidor Web, copias de seguridad.

1.6 Comandos básicos

La estructura básica de un comando en Linux es la siguiente:

```
comando [–opción(es)] [argumento(s)]
```

Al inicio se teclea el nombre del comando que se quiere ejecutar, a continuación las opciones y al final los argumentos.

Las *opciones* modifican el funcionamiento del comando, generalmente se ocupan para agregar información a la salida del comando o aumentar el potencial del comando. No son necesarios para la ejecución del comando.

Los *argumentos* son requisito para la ejecución del comando (en la mayoría de los comandos), si no damos el argumento apropiado para la ejecución del comando simplemente no se ejecutará y el sistema enviará un mensaje de error.

1.6.1 Reconocimiento del servidor y del sistema

A continuación se describen brevemente los comandos más importantes para el reconocimiento del servidor y del sistema.

1.6.1.1 hostname

Este comando muestra el nombre del servidor.

Sintaxis: ***hostname*** [-opciones]

Opciones:

```
-i    Esta opción muestra la dirección IP que tiene asignada
      el servidor de trabajo.
-d    Muestra el dominio del servidor.
-f    Muestra un listado completo incluyendo el nombre del
      servidor y su dominio.
```

1.6.1.2 uname

Este comando nos indica el tipo de Linux que se está utilizando.

Sintaxis: ***uname*** [-opciones]

Opciones:

```
-n    Indica el nombre del servidor.
-r    Muestra la versión del kernel.
-m    Indica la arquitectura del procesador.
-a    Muestra todos los datos antes mencionados.
```

1.6.2 Creación y gestión de archivos y directorios

A continuación se describen los comandos más importantes para la gestión de archivos y directorios.

1.6.2.1 ls

Muestra el listado de un directorio o archivo, los resultados se muestran ordenados alfabéticamente.

Sintaxis: ***ls*** [-opciones] [archivos]

Opciones:

```
-l    Formato de salida largo, con más información.
-a    Con esta opción se muestran también archivos y
      directorios ocultos.
-R    Lista recursivamente los subdirectorios.
```

1.6.2.2 cd

Este comando permite cambiar de directorio de trabajo.

Sintaxis: ***cd*** [ruta_directorio]

Si se usa el shell *BASH*, (que se instala por default en los sistemas Linux), simplemente tecleando *cd* volvemos a nuestro directorio HOME.

1.6.2.3 pwd

Imprime el directorio actual de trabajo, es decir donde se encuentra posicionado dentro del sistema de directorios indicándonos la ruta absoluta.

Sintaxis: ***pwd***

1.6.2.4 touch

Actualiza la fecha de modificación de un archivo, o crea un archivo vacío si el archivo pasado como parámetro no existe.

Sintaxis: ***touch*** [-opciones] archivo...

Opciones:

-a Cambia la hora de acceso del archivo.

1.6.2.5 mkdir

Permite la creación de directorios, es posible crear varios directorios en la misma instrucción.

Sintaxis: ***mkdir*** [-opciones] directorio...

1.6.2.6 cp

El comando **cp** sirve para copiar archivos, se puede copiar un sólo archivo, múltiples archivos, copiar un archivo a un directorio específico o copiar directorios completos.

```
Sintaxis:    cp [-opciones] archivo_fuente archivo_destino
             cp [-opciones] archivo_fuente ruta_destino

Opciones:

-i    Interactivo. Pide la confirmación  para sobrescribir el
      archivo destino si este ya existe.
-r    Recursivo. Copiará el directorio y todos sus archivos,
      incluyendo subdirectorios y sus archivos al destino.
```

1.6.2.7 rm

Este comando sirve para borrar archivos, permite borrar un solo archivo o varios utilizando *expresiones regulares*[*].

```
Sintaxis: rm [-opciones] archivo…

Opciones:

-r    Esta opción hace un borrado de forma recursiva, se debe
      tener mucho cuidado ya que se borrará el directorio
      especificado y todo su contenido.
-f    La opción –f fuerza a que los elementos sean borrados
      aunque no se tengan permisos sobre ellos.
```

1.6.2.8 mv

Este comando tiene dos funciones básicas, la primera es mover archivos o directorios de un lugar a otro, y la segunda es renombrar archivos o directorios.

```
Sintaxis:    mv [-opciones] archivo ruta_destino
             mv [-opciones] archivo nuevo_nombre
```

1.6.3 Trabajo con archivos

Estos son los comandos más comunes en Linux, para el trabajo con archivos.

1.6.3.1 cat

El comando *cat* permite ver el contenido de uno o más archivos.

[*]La utilidad más obvia de una **expresión regular** es la de describir un conjunto de cadenas, lo que resulta de gran utilidad en editores de texto y aplicaciones para buscar y manipular textos. Muchos lenguajes de programación admiten el uso de expresiones regulares con este fin.

Sintaxis: ***cat*** [-opciones] archivos…

Opciones:

-n Utilizando esta opción el comando imprime el contenido del(os) archivo(s) numerando las líneas en forma ascendente.

1.6.3.2 more

Despliega el contenido de un archivo por pantallas, para avanzar una pantalla se presiona la barra espaciadora o *enter* si se quiere avanzar una línea.

Sintaxis: ***more*** [-opciones] archivos…

1.6.3.3 head

Muestra las primeras 10 líneas de un archivo.

Sintaxis: ***head*** [-opciones] archivo

Opciones:

-n Esta opción le indica al comando que despliegue las primeras “n” líneas, en lugar de 10.

1.6.3.4 tail

Muestra las últimas 10 líneas de un archivo.

Sintaxis: ***tail*** [-opciones] archivo

Opciones:

-n Esta opción le indica al comando que despliegue las últimas “n” líneas, en lugar de 10.

1.6.3.5 sort

Ordena el contenido de un archivo por líneas en orden alfabético.

Sintaxis: ***sort*** [-opciones] archivo…

Opciones:

-r Ordena el contenido de un archivo en forma inversa alfabéticamente.

1.6.3.6 wc

Cuenta las líneas, palabras y caracteres contenidos en un archivo.

Sintaxis: ***wc*** [-opciones] archivo…

Opciones:

-l Cuenta únicamente las líneas
-w Cuenta únicamente las palabras
-c Cuenta únicamente los caracteres

1.7 Rutas absolutas y relativas

Para explicar el concepto de rutas absolutas y relativas, se parte de la siguiente estructura de directorios:

```
/
`-- home
    `-- soporte
        |-- edgar
        |   |-- informes.txt
        |   `-- proyectos.txt
        |-- juan
        `-- lulu
```

Las ***rutas absolutas*** especifican la ruta que conduce a un directorio o archivo, empezando por el directorio raíz "/"en la parte superior de la estructura de árbol.

Ejemplo:

La ruta absoluta para el archivo `informes.txt` que está dentro del directorio `edgar` quedaría de la siguiente forma:

```
/home/soporte/edgar/informes.txt
```

Una ***ruta relativa*** se puede utilizar como un acceso directo a la ubicación de los archivos y directorios. Las rutas relativas especifican directorios y archivos comenzando por el directorio de trabajo actual (en lugar del directorio raíz).

Ruta relativa	Significado
.	Directorio actual
..	El directorio primario (el directorio que está por encima del directorio actual.
../..	Dos directorios por encima del directorio actual.

Ejemplo:

Supongamos que se está en el directorio **edgar**, y se quiere listar los archivos que hay en directorio **home** utilizando una ruta relativa. La instrucción quedaría de la siguiente forma:

```
ls ../../../home/
```

Se pueden utilizar rutas relativas y absolutas en los comandos que se han visto hasta ahora.

1.8 Atributos de los archivos

Dentro del sistema operativo se puede encontrar diferentes tipos de archivos, los cuales podemos distinguir por el primer ***bit*** de sus propiedades. Este ***bit*** nos indicará si se trata de un archivo o un directorio, si es un archivo nos indica su tipo.

Los atributos se componen de 10 bits, el primer bit indica el tipo de elemento que está listado (archivo o directorio), los siguientes 9 bits representan los tipos de permisos asignados a cada elemento listado.

1.8.1 Algunos tipos de archivos

d Indica que el elemento es un directorio.
_ Archivo ordinario.
l El elemento listado es una liga suave.
c Archivo de tipo caracter (Archivos de acceso de hardware mediante un bit como el mouse).
b Archivo de bloque (acceso por bloques en paralelo como la impresora).

1.8.2 Permisos

Existen tres tipos de permisos que se aplican tanto a los archivos como a los directorios, ellos son:

r lectura
w escritura

x ejecución

Los permisos se encuentran asignados en 3 bloques, los cuales representan al usuario o dueño del elemento listado, al grupo al que pertenece el usuario y a los otros usuarios del sistema.

Ejemplo:

```
-rwxr-xr-x  1 becpfe   becarios   11079 Oct 30  2004 a.out
drwxr-xr-x  2 becpfe   becarios     512 Sep  4 00:31 ejemplos
-rw-------  1 becpfe   becarios  470638 Oct 30  2004 outguess-0.2.tar.gz
-rw-r--r--  1 becpfe   becarios      16 Oct 30  2004 palabra.txt
```

1.8.2.1 chmod

El comando *chmod* permite cambiar los permisos de archivos y directorios, existen dos métodos para poder hacerlo: el *simbólico* y el *octal*.

Sintaxis: ***chmod*** `[-opciones] método archivo...`

Si se quiere usar el ***método simbólico***, se debe considerar lo siguiente:

- **u** Indica que se modificarán los permisos del usuario dueño del elemento.
- **g** Indica que se modificarán los permisos asignados al grupo.
- **o** Indica que se modificaran los permisos de los usuarios que no pertenecen al grupo (otros).
- **a** Indica que afectará los permisos para todos los usuarios.

Ejemplo:

```
chmod u+r,g-rx,o+r hola
```

Descripción:

En este ejemplo al archivo *hola* se le agrega el permiso de *lectura* a *usuario* (*u+r*), se le quitan los permisos de lectura y ejecución a *grupo* (g-rx), finalmente se le agrega el permiso de *lectura* a *otros* (o+r).

Con el ***método octal*** se tiene que usar la siguiente tabla:

Asignación	Simbólico	Permisos
0	*_ _ _*	Ninguno
1	*_ _ x*	Ejecución
2	*_ w _*	Escritura
3	*_ w x*	Escritura y ejecución
4	*r _ _*	Lectura
5	*r _ x*	Lectura y ejecución
6	*r w _*	Lectura y escritura
7	*r w x*	Lectura, escritura y ejecución

Ahora solo resta asignar un número (correspondiente a los permisos) para *usuario*, *grupo* y *otros*.

Ejemplo:

```
chmod 754 hola
```

Descripción:

En este ejemplo se le asignan los permisos de *lectura*, *escritura* y *ejecución* (7) a *usuario, lectura* y *ejecución* (5) a *grupo* y finalmente *lectura* (4) a *otros*.

1.9 Redireccionamiento

En Linux existe una salida estándar que es el monitor y una entrada estándar que es el teclado. Las instrucciones que se quieren ejecutar se indican a través del teclado, de la misma forma todo resultado que arroje un comando será mostrado por *default* en pantalla, a no ser que se le indique otra cosa: un "*redireccionamiento*".

1.9.1 Redireccionamiento de salida

El caracter *">"* permite redireccionar la salida de un programa hacia un archivo.

Ejemplo:

```
ls -l  > miLista
```

Con esta instrucción el resultado del comando *ls –l* no se imprime en pantalla, si no en el archivo llamado *miLista.*

1.9.2 Redireccionamiento de entrada

El caracter "<" permite redireccionar un archivo hacia un programa, es decir, tomar los datos del archivo como argumento.

Ejemplo:

```
mail edgar < mensaje.txt
```

En este ejemplo, se le envía el contenido del archivo *mensaje.txt* por correo al usuario *edgar*.

1.9.3 Redireccionamiento de un programa a otro

El caracter "|" funciona como un filtro, manda la salida de un programa hacia la entrada de otro. Enlaza la salida de un programa con la entrada de otro.

Ejemplo:

```
cat texto | more
```

En este caso la salida del comando *cat*, la toma como entrada el comando *more*, el resultado es el despliegue del archivo *texto* por pantallas.

1.9.4 Redireccionamiento no destructivo o de adición

Utilizando ">>" se podrá redireccionar la salida de un programa hacia un archivo, si el archivo no existe lo crea, pero si el archivo existe el resultado del comando lo agrega al final del archivo existente sin eliminar el contenido del archivo.

Ejemplo:

```
ls >> milista
```

1.9.5 Redireccionamiento condicionado

Existe una salida estándar de verdadero y una de falso, a la salida de verdadero le corresponde el número 1 y a la salida de falso le corresponde el número 2, de tal forma que si en algún comando no se cumple una condición en lugar de que el error se imprima en pantalla puede ser redireccionado hacia un archivo.

Ejemplo:

```
ls –R /etc 1>milistado 2> /dev/null
```

En esta instrucción, la salida de verdad se manda al archivo *milistado* mientras que la salida de error esta direccionada al archivo */dev/null.*

1.10 Ligas

Existen dos tipos de ligas, una de ellas es la de tipo suave y la otra es de tipo dura. A continuación se explica en que consiste cada una.

1.10.1 Liga suave

La liga de tipo suave es un vínculo hacia un archivo, pero la liga es de tamaño muy pequeño ya que solo hace referencia al archivo que se está ligando, funciona como un acceso directo y su función generalmente es asociar a un programa de nombre complejo con una liga la cual tiene un nombre más fácil de recordar.

Sintaxis: ***ln*** `–s nom_archivo  nom_liga`

El comando *ln* permite generar las ligas, en este caso con la opción *–s* se indica al sistema que se esta generando una liga de tipo suave.

1.10.2 Liga dura

La liga dura al igual que la liga suave mantiene una relación con un archivo el cual esta ligado, la diferencia es que al comparar el tamaño de la liga con el archivo, ambos tienen el mismo tamaño, al editar el contenido de uno se actualiza automáticamente el otro, pero a comparación de un archivo común es que la liga esta asociada al mismo *inodo*[⊥] del archivo con el que esta ligado.

Sintaxis: ***ln*** `nom_archivo  nom_liga`

En este caso se esta generando una liga dura, al aplicar el comando ln sin opciones, el sistema interpreta el comando como la creación de una liga dura.

[⊥] Un **inodo** o Index Node es un apuntador a sectores específicos de disco duro en los cuales se encuentra la información del archivo, un inodo también contiene información de permisos, propietarios y grupos a los cuales pertenece el archivo.

2. Instalación y administración de Linux

El administrador de sistemas es la 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 de forma que estén disponibles para ser utilizados por los usuarios.

2.1 Importancia de la administración

- Proporcionar un ambiente seguro, eficiente y confiable.
- Brindar un funcionamiento confiable del sistema.
- Se divide el trabajo entre varios administradores, dependiendo del tamaño del sistema.

2.2 Tareas administrativas comunes

- Administración de usuarios.
- Configuración de dispositivos.
- Creación de respaldos.
- Capacitación de usuarios.
- Asegurar el sistema.
- Registrar los cambios del sistema.
- Asesorar a los usuarios.

2.3 Actividades del administrador

- Mantenimiento de claves de usuarios.
- Instalación y mantenimiento de dispositivos.
  - Impresoras
  - Discos
- Unidades de respaldo
- Instalación y actualización de software (comercial y dominio público).
- Configuración de las interfaces de red.
- Administración de los recursos (cpu, memoria y disco).
- Atención a usuarios.
- Monitoreo del sistema.
- Detección de fallas.
- Auditoria e implantación de la seguridad del sistema.
- Planear las actividades.
- Guardar copias de seguridad (jamás modificar sin respaldar previamente).

2.4 Conocimientos de un administrador

- Técnicas de programación.
- Dominio de al menos un lenguaje de programación.

- Funcionamiento del sistema operativo.
- Técnicas de administración del sistema operativo.
- Conocimientos básicos de hardware y mantenimiento de dispositivos.
- Comprensión profunda sobre redirección, tuberías, procesamiento en segundo plano, etc.
- Manejo de *Vi*[*].
- Programación en shell.
- Utilerías del sistema
    - Básicas: *cut, sort, paste, diff, comm, tail, head, grep, egrep, compress*, etc.
    - Control de tareas: *at, crontab*.
- Herramientas de programación
    - *Lenguaje C*
    - *Shell*
    - *AWK*
    - *PERL*
- Documentación
    - En línea (*man, apropos, info*).
    - Impresa (libros, manuales)
    - Internet.
- Hardware del equipo
    - Características, modelo, capacidad, etc.
    - Ubicación física.
- Mantener canales de comunicación con los usuarios.
    - *news*
    - *wall*
    - *write*
    - *mail*
    - *Web*
- Establecer políticas.
- Apertura de cuentas.
- Horas de mantenimiento.
- Responsabilidad de los respaldos.
- Borrado de archivos temporales.
- Cuotas de disco.
- Seguridad del sistema.

2.5 Instalación de Linux

La instalación de Linux puede variar dependiendo de la distribución con la que se quiera trabajar. En el diplomado se trabajó con la versión 9.0 de la distribución *Slackware*, esta distribución no requiere de un sistema extremadamente potente para ejecutarse.

[*] **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.

Los requerimientos *mínimos* para instalar y ejecutar Slackware son:

- Procesador 386.
- 16 MB en RAM.
- 5 MB de espacio libre en disco duro.
- Unidad de 3.5".
- Hardware adicional, tarjeta de video para ejecutar X Windows y una tarjeta de red por si se desea tener acceso a algún tipo de red.

Antes de instalar Slackware se debe seleccionar las series de paquetes que se van a instalar.

Linux Slackware es una de las distribuciones de Linux más antiguas, contiene un conjunto de paquetes que se reparten en series y comienzan desde la letra A hasta la Z.

Series

A	Es la base, contiene suficiente software para levantar y ejecutar Slackware, contiene algunos editores de texto y programas de comunicaciones.
AP	Varias aplicaciones que no requieren del sistema X.
D	Herramientas de desarrollo de programas: compiladores, depuradores, intérpretes y los manuales n.
DES	Incluye la función crypt( ) de libc de GNU.
E	GNU Emacs, es tan grande que requiere su propia serie.
F	Contiene FAQ's, HOWTO's y otro tipo de documentación.
GTK	Contiene el ambiente de escritorio de GNOME, biblioteca de widgets de GTK, y el UIMP.
K	El código fuente del núcleo de Linux.
KDE	Contiene el ambiente de trabajo de escritorio de KDE.
N	Contiene programas para configuración de una red. Demonios, programas de correo, telnet, programas de lectura de noticias, etc.
T	Contiene el sistema y formato de documentos teTeX.
TCL	Contiene las herramientas el lenguaje de comandos, elt Tk, el TclX y el TkDesk.
U	Contiene paquetes de programas diseñados específicamente para trabajar solamente en sistemas de UltraSPARC.
X	Contiene la base del sistema *X Windows*[*].
XAP	Contiene aplicaciones X que no son parte de un ambiente de escritorio importante. Por ejemplo. Ghostscript y Netscape.
XD	Contiene Bibliotecas, kitde la conexión del servidor, y ayuda de PEX. Para el desarrollo de X11.
XV	Contiene juegos (una colección de juegos de BSD)

[*] El sistema **X Window** (*Window*, sin la 's' final) fue desarrollado a mediados de los años 80 en el MIT para dotar de una interfaz gráfica a los sistemas Unix.

2.5.1 Proceso de instalación de Slackware 9.0

Existen diferentes formas de instalar Slackware, dependiendo de las características del equipo se pueden elegir entre alguna de estas. Las formas de instalación son las siguientes:

- CD-ROM
- Discos de arranque
- Disco de red
- Disco PCMCIA

Para explicar el proceso de instalación de Slackware se utilizará el tipo de instalación más frecuente, desde CD-ROM (se debe especificar en el BIOS* el arranque desde CR-ROM).

Al arrancar la instalación aparecerá una pantalla de bienvenida al programa de instalación de Linux, en la parte inferior de esta pantalla aparecerá el indicador *boot.*

Esta pantalla contiene información sobre algunas opciones de arranque, se eligió la opción más sencilla y solo se tecleó *enter,* de esta forma comenzará la instalación.

Después de arrancar la instalación, Linux detectará la mayor parte del hardware que está instalado en nuestro equipo.

El primer paso de la instalación es elegir la configuración de nuestro teclado.

El siguiente paso es ingresar al programa de instalación (setup) de la siguiente forma:

```
Slackware login: root
```

En seguida se obtendrá un *prompt* donde se podrá ejecutar una serie de comandos básicos para la instalación de Linux.

Antes de crear los paquetes se deben crear las particiones necesarias para la instalación.

Se ejecutará el comando *fdisk,* para poder trabajar con las participaciones de la siguiente forma:

* El sistema básico de entrada/salida **Basic Input-Output System (BIOS)** es un código de interfaz que localiza y carga el sistema operativo en la RAM. Proporciona la comunicación de bajo nivel, y el funcionamiento y configuración del hardware del sistema que, como mínimo, maneja el teclado y proporciona salida básica durante el arranque.

Sintaxis: ***fdisk*** /dev/hda

Opciones:

p	Despliega la tabla de particiones actual.
m	Despliega en la pantalla la ayuda.
d	Elimina una partición.
n	Crear una nueva partición.
t	Cambia el sistema de identificación de la partición.
q	Sale de fdisk sin guardar los cambios.
w	Escribe los cambios en el disco y sale de fdisk.

Antes de particionar el disco duro, es importante tomar en cuenta las siguientes consideraciones.

- En Linux es habitual tener más de una *partición*[≈], el área de SWAP[*] y el sistema de archivos Linux, esto es en la mayoría de los casos. Muchas personas instalan Windows y Linux en el mismo disco duro, en particiones independientes lo que permite flexibilidad y compatibilidad unidireccional.
- Para evitar problemas de seguridad y facilitar el trabajo del administrador, se crean particiones independientes para los sistemas de archivos principales del sistema.
- **NO** se debe colocar el sistema de archivos raíz y de usuario en la misma partición Linux.
- Tener en una misma partición todo Linux dificulta la tarea de los administradores de sistemas.
- Se debe crear una partición independiente para cada uno de los sistemas principales de archivos.
- Los sistemas de archivos importantes que deben estar en particiones independientes son: */ (raíz), /var y /usr* desde el punto de vista administrativo.

Antes de comenzar a trabajar con *fdisk* se necesita saber cuantas particiones se tienen en el disco duro y cuanto espacio están ocupando (para ver las particiones, se teclea la letra **p**).

Para agregar una partición se teclea la letra **n,** *fdisk* preguntará si se quiere crear una partición primaria o extendida, *todas las particiones en donde se va a instalar algún tipo de sistema operativo* siempre deben ser primarias, así que tecleamos **p** para una partición primaria.

[≈] La **partición** de disco duro es la creación de divisiones lógicas en un disco duro que permite aplicar el formato lógico de un sistema operativo específico.

[*] La **swap** es un espacio reservado en el disco duro para poder usarse como una extensión de memoria virtual de tu sistema. Es una técnica utilizada desde hace tiempo para hacer creer a los programas que existe mas memoria RAM de la que en realidad existe.

Ahora *fdisk* preguntará por el número de una partición (1-4), dependiendo del número de particiones que se tengan, enseguida pedirá el primer cilindro para la nueva partición, solo se teclea *enter* y preguntará por el último cilindro, el cual se puede especificar en megas de la siguiente manera:

```
+tamañoM
```

Donde tamaño es la cantidad en megas de la partición, ejemplo:

```
+2000M     partición de 2Gb.
+3500M     partición de 3.5Gb.
```

De acuerdo con el tamaño de la partición se debe indicar de este modo.

Ya se tiene la primera partición, ahora se debe crear la partición *swap*.

La forma de crear esta partición es la misma. Cuando se hayan creado las dos particiones anteriores aparecerán como Linux nativa, se debe cambiar el tipo de la segunda partición que es la partición *swap*.

Se oprime la letra **t** para cambiar el identificador, se indica el número de la partición (1-4) que se va a modificar y se selecciona el tipo de identificador de una lista oprimiendo la tecla **l**, se elige el identificador, que es el *82*, se teclea y se presiona *enter*.

Ahora se tienen las dos particiones, se guardan los cambios con la tecla **w** y se continua con la instalación.

Una vez realizadas las particiones del sistema, para arrancar el programa de instalación se teclea la palabra ***setup*** y aparecerá la pantalla de inicio de instalación. El programa de instalación nos llevará de la mano en éste proceso.

Se puede mover a través del menú con las teclas de navegación y con la tecla Tab.

Se selecciona la tercera opción *add swap*, el programa detectará la partición swap y preguntara si se quiere activarla y darle formato, se elige ***OK*** y se presiona *enter*. El programa de instalación guiará fácilmente en este proceso.

Cuando el programa de instalación detecta particiones que contienen otro sistema operativo, pide un punto de montaje para que sea accesible desde Linux, solo se debe indicar la ruta en donde estará montada la partición y el nombre del directorio.

```
/windows o /win98
```

Cuando aparezca la opción de instalar LILO[*], éste se alojará en el MBR[†] (Master Boot Record).

Al terminar la instalación se debe reiniciar el equipo, seleccionar EXIT del menú y se oprimen las teclas *control + alt + supr.*

Cuando el sistema arranque nuevamente seleccionamos Linux del menú de arranque.

2.6 Dar de alta o baja el sistema

Linux a diferencia de otros sistemas operativos, necesita tiempo para cerrar los procesos abiertos, guardar los datos no guardados en el disco duro.

Para apagar el sistema Linux, se debe utilizar el comando ***shutdown.***

2.6.1 shutdown

Este comando está especialmente diseñado para desconectar Linux de forma segura.

Sintaxis: ***shutdown*** –opciones

Opciones:

-c Se utiliza para cancelar un apagado que ya estaba programado.
-h Se utiliza para forzar una detención de todo el sistema tras apagarse el sistema.
-k Se utiliza para simular un apagado.
-r Se utiliza para forzar un reinicio tras apagarse el sistema.
-t Se utiliza para establecer el tiempo en segundos, antes de que *shutdown* realice su tarea.

Ejemplo:

```
shutdown –r now
```

[*] **LILO** permite tener un sistema de arranque múltiple: Windows, MS-DOS, Linux, etc. Este programa se ubica en el sector de arranque de su disco y le permite seleccionar la partición sobre la cual desea arrancar. El archivo de configuración LILO se encuentra generalmente en **/etc/lilo.conf**. Las distribuciones permiten generar un archivo automáticamente.

[†] **Master boot record** es la traducción del inglés de "registro principal de arranque" (acrónimo **MBR**), es un sector de 512 bytes al principio del disco duro que contiene una secuencia de comandos necesarios para cargar un sistema operativo. Es decir, es el primer registro del disco duro, el cual contiene un programa ejecutable y una tabla donde están definidas las particiones del disco duro.

Descripción:

Con esta instrucción se apagará inmediatamente el sistema y reiniciará:

Ejemplo:

```
shutdown –r 12:24
```

Descripción:

En este ejemplo se expresa en horas, de una forma más concreta, pero el valor del tiempo también se puede expresar de forma en minutos (**`shutdown –r + minutos`**).

2.7 Cuentas de usuario

Al usuario *root* se le otorga todo el poder administrativo, *root* controla a los usuarios individuales a los grupos y a los archivos.

El sistema mantiene una cierta cantidad de información acerca de cada usuario. El archivo *etc/passwd* contiene la información acerca de los usuarios, cada línea del archivo contiene información acerca de un único usuario.

El formato de cada línea es el siguiente:

```
nombre:clave_encriptada:UID:GID:nombrecompleto:dir_inicio:intérprete
```

Donde:

`nombre`	Es el nombre de usuario, el identificador único dado a cada usuario del sistema.
`clave`	El sistema almacena una clave encriptada* del usuario.
`UID`	El *user id* es un número único dado a cada usuario del sistema.
`GID`	El *group id* es el identificador del grupo del usuario por defecto.
`nombre completo`	Es el nombre completo del usuario.
`directorio inicial`	Es el directorio en el que se coloca inicialmente al usuario en tiempo de conexión. Cada usuario debe tener su propio directorio inicial, normalmente situado bajo */home*.
`intérprete`	Es el intérprete de comandos que es arrancado para el usuario en tiempo de conexión.

* **Encriptación:** Procedimiento de seguridad para codificar mensajes y programas, de modo que no puedan ser leídos o copiados si se desconoce la clave de encriptación.

2.8 Agregando usuarios

Existen varias formas de añadir un usuario: utilizando herramientas gráficas, modificando el archivo */etc/passwd* manualmente o utilizando el comando *useradd* que se describe a continuación.

2.8.1 useradd

El comando *useradd* permite agregar un usuario. Cuando es invocado sin la opción –D el comando useradd crea una nueva cuenta de usuario usando los valores especificados en la línea de comando y los valores por defecto del sistema.

Sintaxis: ***useradd*** [-opciones] nombre_usuario

Opciones:

-c	Comentario
-d	Directorio home
-e	Fecha de expiración
-g	Grupo inicial
-p	Password
-s	Shell
-u	UID

Ejemplo:

```
useradd -c usuario_muestra -u 600 -g 501 -s /bin/bash -d
/export/home/nuevousuario nuevousuario
```

2.9 Borrando usuarios

Si se quiere borrar un usuario puede hacerse con el comando *userdel* que se describe a continuación.

2.9.1 userdel

El comando *userdel* elimina un usuario y los archivos relacionados.

Sintaxis: ***userdel*** [-opciones] nombre_usuario

Opciones:

-r	Los archivos en el directorio home del usuario serán eliminados junto con su directorio home.

2.10 Grupos

Cada usuario pertenece a uno o más grupos, cada archivo tiene un "grupo propietario" y un conjunto de permisos de grupo que define de qué forma pueden acceder al archivo los usuarios del grupo.

El archivo *etc/group* contiene información acerca de los grupos. El formato de cada línea es el siguiente:

```
nombre_grupo:clave:GID:otros_miembros
```

Si se quiere agregar un nuevo grupo, puede hacerse con el comando *groupadd* que se describe a continuación:

2.10.1 groupadd

El comando groupadd crea un nuevo grupo.

Sintaxis: ***groupadd*** [-opciones] nombre_grupo

Opciones:

-g Valor numérico para GID.

Muchas de las primeras versiones de Linux almacenaban las contraseñas de los usuarios en */etc/passwd* lo que no resultaba seguro, ya que ese archivo es y debe ser legible, De ahí que cualquier usuario pueda ver el contenido de */etc/passwd*. Las contraseñas de Linux están encriptadas, de ésta manera aunque */etc/passwd* fuera público la contraseña era ilegible para cualquier usuario curioso.

La base de datos de contraseñas es el archivo *shadow* que se encuentra en */etc.*

2.11 Dispositivos y Linux

Las distribuciones de Slackware Linux incluyen un kernel reciente con todos los módulos compilados y puede detectar hardware en el momento del arranque, de tal forma que un administrador o un usuario en general no deba encargarse de esto.

Sin embargo puede haber algunos casos excepcionales:

- Si se compra hardware diseñado para Windows o que no tenga especificaciones abiertas puede ser difícil usarlo en Linux.

- Si se compra hardware de reciente generación para el cual aún no haya controladores para Linux (o si su distribución de Linux no es la más reciente).

2.11.1 Disquetes

Aunque hay varios dispositivos para unidades de disquete (dependiendo del formato), los dispositivos */dev/fd0*, */dev/fd1*, etc., autodetectarán el formato. Para acceder a un disquete pueden usarse estos comandos: *mount*, *umount*.

Si se emplea *mount* es recomendable agregar al archivo */etc/fstab*:

```
/dev/fd0  /floppy  auto  defaults,user,noauto   0  0
```

2.11.2 Unidades de CD-ROM y DVD

Una vez que se configure la unidad de CD o DVD se podrán montar CDs o con un programa apropiado podrá escuchar CDs de audio. Los controladores para los tipos más comunes de unidades de CD y DVD también soportan CDs multisesión y algunos soportan lectura de datos digitales.

El kernel 2.2.x como la 2.4.x soportan las unidades de CD-ROM y DVD más estándares IDE/ATAPI y SCSI así como algunas unidades no tan recientes con sus propias interfaces.

3. Programación en Shell

La programación en shell es una de las herramientas más apreciadas por los administradores y muchos usuarios de Linux, ya que nos permite automatizar tareas complejas y repetitivas y ejecutarlas con un simple llamado al script* o hacerlo automáticamente a determinadas horas.

Un shell no solo es un intérprete de comandos, también es un auténtico lenguaje de programación y como tal, incorpora estructuras de control de flujo, sentencias de asignación, funciones etc.

Los scripts de shell no necesitan ser compilados como ocurre en otros lenguajes de programación. El propio shell los interpreta línea por línea.

3.1 Primer programa de shell

Para crear este tipo de programas, lo primero que se debe hacer es elegir el nombre del script. A continuación se utilizará un editor de texto (*vi, pico, emacs*, etc.) y se introduce un par de líneas correspondientes a dos comandos en Linux (*who* y *date*). Finalmente se le otorgan los permisos de ejecución al script.

Ejemplo:

```
#primerScript.sh
#Esto es un comentario
who
date
```

Se le cambian los permisos de la siguiente forma:

```
chmod 755 primerScript.sh
```

Por último se ejecuta con alguna de las siguientes instrucciones:

```
sh primerScript.sh  o ./primerScript.sh
```

Estos pasos son los que se deben seguir para crear y ejecutar un script de shell.

3.2 Paso de parámetros a un programa de shell

Un script puede recibir parámetros desde la línea de comandos, a estos parámetros se les conoce como parámetros de posición y se pueden usar

* Los programas escritos mediante lenguajes interpretados se suelen llamar **scripts**.

dentro de un programa de shell como cualquier otra variable. Para saber su valor se debe utilizar el símbolo *$*.

$0 Representa al parámetro cero o nombre del programa.
$1 Representa al parámetro uno.
…
$9 Representa al parámetro nueve.

Solo se pueden usar los parámetros de posición para referenciar hasta 9 argumentos.

Ejemplo:

```
#parametros.sh
#Este shell script visualiza los 4 primeros parámetros
echo Parámetro 0 = $0
echo Parámetro 1 = $1
echo Parámetro 2 = $2
echo Parámetro 3 = $3
```

Descripción:

Este script visualiza los 4 primeros parámetros que se pasan desde la línea de comando. El comando *echo* permite desplegar un mensaje en pantalla.

Salida:

```
[epena@hermes moduloIII]$ sh parametros.sh uno dos tres
Parámetro 0 = parametros.sh
Parámetro 1 = uno
Parámetro 2 = dos
Parámetro 3 = tres
```

3.3 Algunas variables especiales del shell

Dentro del shell existen variables con significados especiales que pueden ser de gran utilidad. Algunas de las cuales son:

Guarda el número de argumentos de la línea de comandos (excluyendo el nombre del programa).
***** Guarda la cadena de argumentos entera (excluyendo el nombre del programa).
? Guarda el código de retorno de la ultima orden ejecutada (0 si no hay error y distinto de 0 si hay error).

Ejemplo:

```
#variables.sh
echo La variable \# vale: $#
echo La variable \* vale: $*
cp
echo La variable \? vale: $?
```

Este script visualizará las variables *"#", "*"* y *"?"* (para evitar que un caracter especial sea interpretado por el shell, se debe anteponer el caracter *"\"*). La variable *"?"* tomará un valor distinto de cero, ya que el comando *cp* se ejecutará con errores.

Salida:

```
[epena@hermes moduloIII]$ sh variables.sh uno dos tres
La variable # vale: 3
La variable * vale: uno dos tres
cp: falta un fichero como argumento
Pruebe `cp --help' para más información.
La variable ? vale: 1
```

3.4 shift

Este comando se utiliza para desplazar los argumentos, de manera que *$2* pasa a ser *$1, $3* pasa a ser *$2*, y así sucesivamente (esto si el desplazamiento n es igual a 1).

Sintaxis: `shift n`

Ejemplo:

```
#shitf1.sh
echo \$1 vale: $1
echo \$2 vale: $2
echo \$3 vale: $3
shift 2
echo Ahora \$1 vale: $1
echo Ahora \$2 vale: $2
echo Ahora \$3 vale: $3
```

Descripción:

Este script desplazará dos lugares los parámetros posicionales, es decir, *$5* pasará a ser *$3, $4* pasará a ser *$2* y *$3* pasará a ser *$1*. Los argumentos iniciales *$1* y *$2,* se pierden después del desplazamiento. Evidentemente este desplazamiento afecta también a las variables # y *.

Salida:

```
[epena@hermes moduloIII]$ sh shitf1.sh uno dos tres cuatro
cinco
$1 vale: uno
$2 vale: dos
$3 vale: tres
Ahora $1 vale: tres
Ahora $2 vale: cuatro
```

3.5 read

El comando *read* se utiliza para leer información escrita en la línea de comandos de forma interactiva.

Sintaxis: ***read*** variable(s)

Ejemplo:

```
#read.sh
#La opción -n se emplea para evitar el retorno de carro.
echo -n "Introduce un valor: "
read var
echo "La variable \"var\" tiene el valor: $var"
```

Descripción:

En este ejemplo se leerá una variable desde la entrada estándar, y posteriormente se visualizará el contenido.

Salida:

```
[epena@hermes moduloIII]$ sh read.sh
Introduce un valor: hola mundo
La variable "var" tiene el valor: hola mundo
```

Si existen más variables en el comando *read* que palabras escritas, las variables que sobran por la derecha se asignan a NULL. Si se introducen más palabras que variables, todos los datos que sobran por la derecha se asignan a la última variable de la lista.

3.6 Operadores aritméticos

Estos operadores se utilizan para evaluar operaciones matemáticas. Las operaciones que se pueden realizar son: suma, resta, multiplicación, división entera y cálculo del resto de la división entera (módulo).

Operadores aritméticos:

+	Suma arg2 a arg1
-	Resta arg2 a arg1
*****	Multiplica los argumentos
/	Divide arg1 entre arg2 (división entera)
%	Resto de la división entera entre arg1 y arg2.

3.7 Operadores relacionales

Estos operadores se utilizan para comparar dos argumentos (pueden ser palabras). Si el resultado de la comparación es cierto, el resultado es uno (1); si es falso, el resultado es cero (0).

Operadores relacionales:

= ¿Los argumentos son iguales?
!= ¿Los argumentos son diferentes?
> ¿El arg1 es mayor que arg2?
>= ¿El arg1 es mayor o igual que arg2?
< ¿El arg1 es menor que arg2?
<= ¿El arg1 es menor o igual que arg2?

Los símbolos > y < tienen significado especial para el shell, por lo que deben ser entrecomillados.

3.8 Operadores lógicos

Los operadores lógicos se utilizan para combinar múltiples comparaciones en una expresión condicional. Un operador lógico toma dos argumentos (operandos) cada uno de los cuales es un valor *true* o *false* y devuelve un valor *true* o *false*.

Operadores lógicos:

| OR Lógico. Devuelve *true* si uno de los operadores es *true*.
& "AND Lógico. Devuelve *true* si ambos operadores son *true*.

3.9 expr

El comando *expr* evalúa una expresión. Los argumentos de este comando se toman como expresiones y deben estar separados por un espacio en blanco.

Sintaxis: ***expr*** `arg1 op arg2 [op arg3 ...]`

Ejemplo:

```
#expr1.sh
#Multiplica dos variables leídas desde el teclado
echo -n "Introduce la primera variable: "
read numero1
echo -n "Introduce la segunda variable: "
read numero2
resultado=`expr $numero1 \* $numero2`
echo Resultado = $resultado
```

Descripción:

Este script leerá dos variables (*numero1* y *numero2*). Se realizará la multiplicación de *numero1* y *numero2* y el resultado se asignará en la variable *resultado*. Finalmente se imprimirá el contenido de la variable *resultado*.

Salida:

```
[epena@hermes moduloIII]$ sh expr1.sh

Introduce la primera variable: 10
Introduce la segunda variable: 12
Resultado = 120
```

3.10 test

El comando *test* se usa para evaluar expresiones y generar un valor de retorno; este valor no se escribe en la salida estándar, pero asigna O al código de retorno si la expresión se evalúa como true, y le asigna 1 si la expresión se evalúa como false.

Se puede invocar el comando *test* también mediante *[expresión]*, tanto a la derecha como a la izquierda de expresión debe haber un espacio en blanco, *test* puede evaluar tres tipos de elementos: archivos, cadenas y números.

3.10.1 Evaluación de archivos con *test*

Sintaxis: ***test*** **[-opciones] archivo**

Opciones:

-f Devuelve verdadero (0) si el archivo existe y es un archivo regular (no es un directorio ni un archivo de dispositivo).
-s Devuelve verdadero (0) si el archivo existe y tiene un tamaño mayor que cero.
-r Devuelve verdadero si el archivo existe y tiene permiso de lectura.
-w Devuelve verdadero si el archivo existe y tiene permiso de escritura.
-x Devuelve verdadero si el archivo existe y tiene permiso de ejecución.
-d Devuelve verdadero si el archivo existe y es un directorio.

Ejemplo:

```
[epena@hermes moduloIII]$ test -f /etc/passwd
[epena@hermes moduloIII]$ echo $?
0
```

Descripción:

En este ejemplo se pregunta si existe el archivo */etc/passwd.* El archivo existe, por lo tanto el código de retorno es 0 (verdadero).

3.10.2 Evaluación de cadenas

Sintaxis:
```
[ cadenal  = cadena2 ]
[ cadenal != cadena2 ]
```

Ejemplo:

```
[epena@hermes moduloIII]$ a=palabra1
[epena@hermes moduloIII]$ [ "$a" = "palabra2" ]
[epena@hermes moduloIII]$ echo $?
1
[epena@hermes moduloIII]$ [ "$a" = "palabra1" ]
[epena@hermes moduloIII]$ echo $?
0
```

Descripción:

De esta manera, *test* evaluará si las cadenas son iguales o distintas, regresa un 1 si las cadenas son distintas y un 0 si son iguales.

3.10.3 Evaluaciones numéricas

Sintaxis: `test número relación número`

El comando *test* en evaluaciones numéricas solo trabaja con números enteros.

Operadores relacionales:

-lt Menor que
-le Menor o igual que
-gt Mayor que
-ge Mayor o igual que
-eq Igual a
-ne No igual a

Ejemplo:

```
[epena@hermes moduloIII]$ a=15
[epena@hermes moduloIII]$ test $a -lt 10
[epena@hermes moduloIII]$ echo $?
1
[epena@hermes moduloIII]$ test $a -eq 15
[epena@hermes moduloIII]$ echo $?
0
```

Descripción:

En el primer caso se hace la pregunta ¿15 es menor que 10?, el resultado de esta evaluación es falso y la expresión regresa un 1. En el segundo caso se hace la pregunta ¿15 es igual a 15?, en este caso la expresión resulta verdadera y regresa un 0.

3.11 if

La estructura *if* se utiliza para tomar decisiones a partir de los códigos de retorno. Funciona como en cualquier otro lenguaje de programación.

Sintaxis:

```
if [ condición ]
then
      instrucción 1
else
      instrucción 2
fi
```

Ejemplo:

```
# if.sh
#shell script que muestra el uso de la sentencia de control if-fi.
if test -f /etc/hosts
then
        cat /etc/hosts
else
        echo "El archivo no existe"
fi
```

Descripción:

Si existe el archivo */etc/hosts,* entonces se despliega su contenido, si no existe imprimimos un mensaje diciendo que ese archivo no existe.

Salida:

```
[epena@hermes moduloIII]$ sh if.sh
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1    hermes.cichcu.unam.mx hermes localhost.localdomain
localhost
132.248.9.2  hermes.cichcu.unam.mx  hermes
```

3.12 case

La estructura selectiva *case* controla el flujo del programa basándose en la palabra dada. La palabra se compara en orden, con todas las plantillas.

Cuando se encuentre la primera que corresponde, se ejecuta la lista de instrucciones asociadas, la cual tiene que terminar con dos punto y coma (;;).

Sintaxis:

```
case palabra in
      patrón1)
            instrucción 1;;
      patrón2)
            instrucción 2;;
      patrónN)
            instrucción n;;
esac
```

Ejemplo:

```
#case.sh
#Cortamos los primeros 3 caracteres de la salida del
#Comando date y lo asignamos a la variable dia
dia=`date | cut -c 0-3`
# Evaluamos la variable dia
case $dia in
       lun)      echo Hoy es Lunes;;
      mar)       echo Hoy es Martes;;
      mie)       echo Hoy es Miércoles;;
      jue)       echo Hoy es Jueves;;
      vie)       echo Hoy es Viernes;;
      sab)       echo Hoy es Sábado;;
      dom) echo Hoy es Domingo;;
esac
```

Descripción:

En la variable *dia* se almacena lo que retorna la instrucción *date | cut -c 0-3*, que son las tres primeras letras del día de la semana en español. Finalmente se evalúa el contenido de la variable *dia* en la estructura *case* y se imprime el mensaje correspondiente.

Salida:

```
[epena@hermes moduloIII]$ sh case.sh
Hoy es Lunes
```

3.13 while

La construcción de la estructura repetitiva *while* es la siguiente:

- Se evalúa la *condición*.
- Si el código devuelto por la *condición* es O (verdadero), se ejecuta el *bloque de instrucciones* y se vuelve a iterar.
- Si el código de retorno de la condición es falso, se salta a la primera instrucción que haya después de la palabra reservada *done*.

Sintaxis:

```
while [ condición ]
do
      Bloque de instrucciones
done
```

Ejemplo:

```
# while.sh
i=1
while [ $i -le 10 ]
do
        echo –n "$i "
        i=`expr $i + 1`
done
```

Descripción:

En este ejemplo se incrementará y visualizará el valor del contador *$i* mientras éste sea menor o igual que 10. Para ello, *while* comprueba el código de retorno de la expresión *[$i -le 10]*, y si es cierto, se repite la iteración.

Salida:

```
[epena@hermes moduloIII]$ sh while.sh
1 2 3 4 5 6 7 8 9 10
```

3.14 until

La construcción de la estructura repetitiva *until* es muy similar a la de *while*:

- Se evalúa la *condición*.
- Si el código de retomo de la *condición* es distinto de O (falso), se ejecuta el *bloque de instrucciones* y se vuelve a iterar.
- Si el código devuelto por la *condición* es O (verdadero), se salta a la primera instrucción que haya después de la palabra clave *done*.

Sintaxis:

```
until [ condición ]
do
      bloque de instrucciones
done
```

Ejemplo:

```
# until.sh
until [ "$a" = "hola" ]
do
     echo -n "Introduce una cadena: "
      read a
done
echo "Fin del programa"
```

Descripción:

El bucle *until* se ejecutará hasta que el usuario introduzca la cadena hola. A partir de este momento, la condición devuelve verdadero y se termina el bucle y muestra el mensaje “Fin del programa”.

Salida:

```
[epena@hermes moduloIII]$ sh until.sh
Introduce una cadena: uno
Introduce una cadena: uno
Introduce una cadena: hola
Fin del programa
```

3.15 for

La construcción de la estructura repetitiva *for* funciona de la siguiente manera:

- Se asigna a *variable* la primera cadena de la *lista*.
- Se ejecuta el *bloque de de instrucciones*.
- Se asigna a *variable* la siguiente cadena de la *lista*. Se vuelve a ejecutar *el bloque de instrucciones*.
- Repetir hasta que se hayan usado todas las cadenas.
- Después de que haya acabado el bucle, la ejecución continúa en la primera línea que sigue a la palabra clave *done*.

Sintaxis:

```
for variable in lista
do
     bloque de instrucciones
done
```

Ejemplo:

```
#for.sh
for i in `cat correos.txt`
do
        mail $i < saludo.txt
done
```

Contenido del archivo *correos.txt*

```
trivialidad@hotmail.com
eipena@finanzas.gob.mx
edgar.pena@idev.com.mx
edgaripf@yahoo.com.mx
```

Contenido del archivo saludo.txt

```
HOLA
```

Descripción:

En este ejemplo se enviará como mensaje el contenido del archivo *saludo.txt* a todas las direcciones de correo que se encuentren en el archivo *correos.txt.*

3.16 Funciones

Las funciones dividen grandes trabajos de computación en partes más breves y aprovechan la labor realizada por otras personas en lugar de partir desde cero. Dentro del cuerpo de la función, $1 es el primer argumento dado a la función, $2 el segundo, etc.

Sintaxis:

```
nombre_de_la_función ( )
{
      Bloque de instrucciones de la función
}
```

Ejemplo:

```
#funcion.sh
# Declaración de la función
ayuda( )
{
       echo "NOMBRE"
       echo "  comando - descripción del comando"
       echo "USO"
       echo "  comando [opciones] archivo(s)"
       echo "OPCIONES"
       echo "  -h Texto de ayuda"
}
# Si el primer parámetro que se le pasa es -h o es una cadena
vacía, # entonces llamamos a la función ayuda
if [ "$1" = "-h"  -o "$1" = "" ]
then
      #Llamamos a la función ayuda
       ayuda
fi
```

Descripción:

Es importante a la hora de crear un script tener un archivo de ayuda, regularmente los comandos tienen un texto de ayuda que se invoca tecleando la opción *"-h"* como primer argumento del comando.

Lo que hace este script en la primera parte es declarar una función llamada *ayuda*, que contiene la descripción y forma de utilización del script.

Después se hace una validación, si el primer argumento es "-h" o si el primer argumento no existe se llama a la función ayuda, para darle al usuario la descripción del comando así como las opciones válidas.

Salida:

```
[epena@hermes moduloIII]$ sh funcion.sh
NOMBRE
        comando - descripción del comando
USO
        comando [opciones] archivo(s)
OPCIONES
        -h Texto de ayuda

[epena@hermes moduloIII]$ sh funcion.sh -h
NOMBRE
        comando - descripción del comando
USO
        comando [opciones] archivo(s)
OPCIONES
        -h Texto de ayuda
```

-

4. Introducción a HTML

El HTML es el lenguaje con el que se escriben las páginas Web. Es un lenguaje utilizado por los navegadores para mostrar las páginas Web al usuario.

Este lenguaje permite aglutinar textos, sonidos, imágenes, video y combinarlos a nuestro gusto. El HTML permite la introducción de referencias a otras páginas por medio de los enlaces hipertexto.

4.1 Sintaxis del HTML

El HTML es un lenguaje que basa su sintaxis en un elemento de base al que llamamos *etiqueta*.

Sintaxis:

```
<etiqueta atributo1="valor1" atributo2="valor2"
atributoN="valorN"> Contenido a modificar </etiqueta>
```

Todo lo incluido en el interior de las etiquetas, sufrirá las modificaciones que caracterizan a esta etiqueta. Además a una etiqueta se le pueden incluir atributos ya definidos, que permiten agregarle funcionalidad y tener un mayor control de los resultados producidos por la etiqueta.

4.2 Partes de un documento HTML

La estructura básica de un documento HTML es la siguiente:

```
<html>
<head>
<title>Título del documento</title>
</head>

<body>
     Cuerpo de la página
</body>
</html>
```

Un documento HTML está delimitado por las etiquetas *<html> y </html>.* En el encabezado (limitado por *<head> y </head>)* se colocan las etiquetas de índole informativo, por ejemplo el título de nuestro documento.

El cuerpo del documento está limitado por las etiquetas *<body>* y *</body>.* Aquí se colocará el texto, imágenes, video etc.

4.3 Formato de párrafos en HTML

Las etiquetas <*p*> y </*p*> nos sirven para definir párrafos, introduce un salto de línea y deja una línea en blanco antes de continuar con el resto del documento.

Atributos importantes para <p>:

`align` Este atributo puede tomar los siguientes valores:
Center si queremos alinear el texto al centro.
Left si queremos alinear el texto a la izquierda.
Right si queremos alinear el texto a la derecha
Justify si queremos un texto justificado.

4.4 Formato de texto

HTML proporciona una serie de etiquetas con las cuales se puede dar un aspecto personalizado al texto del documento. Algunas de las más importantes se describen a continuación.

4.4.1 Negritas

Se puede escribir un texto en negritas incluyéndolo dentro de las etiquetas <*b*> y </*b*> o <*strong*> y </*strong*>.

Ejemplo:

```
<b>Esto estará en negritas</b>
```

4.4.2 Itálicas

También en este caso existen dos posibilidades, una corta: <*i*> e </i> (italic) y otra un poco más larga: <*em*> y </*em*>.

Ejemplo:

```
<i>Texto en itálica</i>
```

4.4.3 Subrayado

Para darle el efecto de subrayado a un texto, se debe utilizar las etiquetas <u> y </u>.

Ejemplo:

```
<u>Este texto estará subrayado</u>
```

4.4.4 Subíndices y supraíndices

Este tipo de formato resulta importante en la publicación de textos científicos, por ejemplo para escribir formulas. Las etiquetas empleadas son las siguientes:

<*sup*> y </*sup*> para los supraíndices.
<*sub*> y </*sub*> para los subíndices.

Ejemplo:

```
La <sup>13</sup>CC<sub>3</sub>H<sub>4</sub>ClNOS es un
heterociclo alergeno enriquecido
```

Salida:

La $^{13}CC_3H_4ClNOS$ es un heterociclo alergeno enriquecido

4.5 Color, tamaño y tipo de letra

Para realizar estas modificaciones al tipo de letra, se utilizan las etiquetas <*font*> *y* </*font*> y algunos atributos de esta.

Atributos importantes para <font>:

face Este atributo permite cambiar el tipo de letra. Es importante mencionar que no todos los usuarios pueden disponer de los mismos tipos de letras.

size Permite cambiar el tamaño de la letra, existen 7 niveles de tamaños distintos numerados del 1 al 7.

color Permite definir el color del texto, el valor puede estar dado en hexadecimal o con el nombre de los colores frecuentemente usados como *aqua,green,black, yellow,* etc.

Ejemplo:

```
<font size="4" face="Arial,Verdana" color="#CCCCCC"> hola
</font>
```

Salida:

hola

4.6 Atributos para páginas

Se puede definir atributos como el color de fondo, el color del texto o el color de las ligas para el documento HTML.

Atributos importantes para <body>.

bgcolor	El color de fondo que se puede asignar es el mismo en toda la superficie del navegador.
background	Con este atributo se puede colocar una imagen como fondo de la página. La imagen se repite muchas veces hasta ocupar toda la superficie.
text	Este atributo sirve para asignar el color del texto de la pagina.
link	El color de las ligas que no han sido visitadas se define con este atributo.
vlink	El color de las ligas visitadas se define con este atributo.
alink	Es el color de los enlaces activos. Un enlace está activo en el preciso instante que se pulsa.
leftmargin	Nos sirve para indicar el margen a los lados de la página.
topmargin	Nos sirve para indicar el margen de arriba de la página.

Ejemplo:

```
<body bgcolor="black" text="aqua" link="yellow" vlink="#CCCCCC"
alink="#FFCC00" leftmargin="0" topmargin="0">
```

4.7 Listas

Las listas son utilizadas para citar, numerar y definir objetos. Existen 3 tipos de listas:

- Listas desordenadas
- Listas ordenadas
- Listas de definición

4.7.1 Listas desordenadas

Están delimitadas por las etiquetas <*ul*> y </*ul*>. Para incluir un elemento en la lista desordenada se debe utilizar la etiqueta <*li*>.

Atributos importantes para <li>:

type	Permite colocar un tipo de viñeta diferente para el elemento de la lista. Los valores que puede tomar son: *circle, square y disc.*

Ejemplo:

```
<p>Países del mundo</p>
<ul>
    <li type="circle">Argentina
    <li type="square">Perú
    <li type="disc">Chile
</ul>
```

Salida:

Países del mundo

- Argentina
  - Perú
- Chile

4.7.2 Listas ordenadas

En este caso se usan las etiquetas <*ol*> y su cierre <*/ol*>. Cada elemento será igualmente precedido de su etiqueta <*li*>.

**Atributos importantes para **

type	Los valores que puede tomar este atributo son los siguientes: *1* Para ordenar por números *a* Por letras del alfabeto *A* Por letras mayúsculas del alfabeto *i* Ordenación por números romanos en minúsculas *I* Ordenación por números romanos en mayúsculas
start	Permite comenzar nuestra numeración por un número o letra que no necesariamente tiene que ser el primero de todos.

Ejemplo:

```
<p>Ordenado por letras</p>
<ol type="a">
     <li>Elemento a
     <li>Elemento b
</ol>
<p>Ordenado por números romanos empezando por el 10</p>
<ol type="i" start="10">
     <li>Elemento x
     <li>Elemento xi
</ol>
```

Salida:

Ordenado por letras

a. Elemento a
b. Elemento b

Ordenado por números romanos empezando por el 10

x. Elemento x
xi. Elemento xi

4.7.2 Listas de definición

En este tipo de listas cada elemento es presentado junto con su definición. La etiqueta principal es *<dl>* y *</dl>.* Las etiquetas del elemento y su definición son *<dt>* y *<dd>* respectivamente:

Ejemplo:

```
<p>Diccionario</p>
<dl>
    <dt>Berro
    <dd>Perro árabe
    <dt>Oreja
    <dd>Sesenta minutejos
</dl>
```

Salida:

```
Diccionario

Berro
        Perro árabe
Oreja
        Sesenta minutejos
```

4.8 Enlaces

El atractivo original del HTML radica en la relación de los contenidos de los archivos, introduciendo referencias bajo forma de enlaces que permitan un acceso rápido a la información deseada.

Para colocar un enlace en la página se utiliza la etiqueta *<a>* y *.* En la etiqueta de apertura se debe especificar el destino del enlace.

Atributos importantes de <a>

`href` Permite especificar el destino del enlace.

`target` Indica a un enlace de hipertexto en qué zona se debe visualizar. Los valores que puede tomar este atributo, son los siguientes:

_blank	Para visualizar el enlace en una nueva ventana.
_self	Para visualizar el enlace en la misma zona donde está el enlace de hipertexto.
_parent	Si el enlace de hipertexto se encuentra en un frame, el enlace se visualizará en el documento del nivel superior.

4.8.1 Enlaces internos

HTML permite hacer enlaces a una sección específica de nuestra página de una forma sencilla. Si se quiere crear un enlace que apunte al final de la página, lo primero será colocar nuestro enlace origen.

En segundo lugar, hay que generar un enlace en el destino. Este enlace deberá tener algún nombre para poder distinguirlo de los otros posibles enlaces realizados dentro de la misma página.

Ejemplo:

Enlace origen

```
<a href = "#abajo">Ir abajo</a>
```

Enlace destino

```
<a name = "abajo"></a>
```

4.8.2 Enlaces locales

Se pueden crear enlaces a páginas que estén en nuestro sitio Web. Para hacerlo es necesario especificar en el atributo *href* de la etiqueta <*a*>, la ruta absoluta o relativa del documento destino.

Ejemplo:

```
<a href = "index.html">Inicio del sitio</a>
```

4.8.3 Enlaces remotos

Este tipo de enlaces es muy común y no representa ninguna dificultad. Simplemente se coloca en el atributo *href* de nuestra etiqueta <*a*> la URL* o dirección de la página con la que se quiere enlazar.

Ejemplo:

```
<a href = "http://www.google.com">ir a google.com</a>
```

* **URL** significa Uniform Resource Locator, es decir, localizador uniforme de recurso. El URL es la cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en Internet.

4.8.4 Enlaces a direcciones de correo

Para colocar un enlace dirigido hacia una dirección de correo se coloca *mailto:* en el atributo *href* del enlace, seguido de la dirección de correo a la que se debe dirigir el enlace.

Ejemplo:

```
<a href = "mailto:edgar@idev.com.mx”> edgar@idev.com.mx </a>
```

4.9 Imágenes en HTML

La etiqueta que se utiliza para insertar una imagen es <*img*>. Esta etiqueta no tiene un cierre correspondiente.

Atributos importantes de <img>:

scr	Con *scr* especificamos la ruta de la imagen.
alt	Permite poner un pequeño texto descriptivo de la imagen.
height	Define la altura de la imagen en pixeles.
width	Define el ancho de la imagen en pixeles.
border	Con *border e*specificamos el borde de la imagen en pixeles.
align	Permite alinear la imagen.

Ejemplo:

```
<img src="res.gif" width="28" height="21" border="3" align="left">
```

4.10 Tablas en HTML

Una tabla en un conjunto de celdas organizadas dentro de las cuales se puede alojar distintos contenidos.

Las tablas son definidas por las etiquetas <*table*> y <*/table*>.

Atributos importantes para <table>

align	Alinea horizontalmente la tabla con respecto a su entorno.
background	Permite colocar un fondo para la tabla a partir de un enlace a una imagen.
bgcolor	Da color de fondo a la tabla.
border	Define el número de pixeles del borde principal.
bordercolor	Define el color del borde.
cellpadding	Define, en pixeles, el espacio entre los bordes de la celda y el contenido de la misma.
cellspacing	Define el espacio entre los bordes (en pixeles).
height	Define la altura de la tabla en pixeles o porcentaje.

width	Define la anchura de la tabla en pixeles o porcentaje.

Si se quiere definir un renglón en nuestra tabla, se deben utilizar las etiquetas *<tr>* y *</tr>.*

Atributos importantes para <tr>

align	Alinea el texto de la celda del mismo modo que si fuese el de un párrafo (*left*, *center* y *right* son los valores que puede tomar este atributo).
valign	Permite que el texto aparezca arriba (*top*), en el centro (*middle*) o abajo (*bottom*) de la celda.
bgcolor	Da color de fondo al renglón elegido.
bordercolor	Define el color del borde.

Dentro de cada renglón, habrá diferentes celdas. Cada una de estas celdas será definida por otro par de etiquetas: *<td>* y *</td>*. Dentro de estas etiquetas será donde se coloque el contenido.

Atributos importantes para <td>

background	Permite colocar un fondo para la celda a partir de un enlace a una imagen.
height	Define la altura de la celda en pixeles o porcentaje.
width	Define la anchura de la celda en pixeles o porcentaje.
colspan	Expande una celda horizontalmente.
rowspan	Expande una celda verticalmente.

Ejemplo:

```
<table>
<tr>  <td> Celda 1, renglón 1</td>
      <td> Celda 2, renglón 1</td>
</tr>
<tr>  <td> Celda 1, renglón 2</td>
      <td> Celda 2, renglón 2</td>
</tr> </table>
```

Salida:

Celda 1, renglón 1 Celda 2, renglón 1
Celda 1, renglón 2 Celda 2, renglón 2

Ejemplo 2:

```
<table width="200" border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td width="100" rowspan="2" align="center" valign="middle">
      rowspan=2
    </td>
    <td width="100"> </td>
  </tr>
  <tr>
    <td width="100"> </td>
  </tr>
</table> <br>
<br>
<table width="200" border="1" cellspacing="0" cellpadding="0">
  <tr align="center">
    <td colspan="2">colspan=2</td>
  </tr>
  <tr>
    <td width="100"> </td>
    <td width="100"> </td>
  </tr>
</table>
```

Salida:

<table>
<tr><td rowspan="2">rowspan=2</td><td></td></tr>
<tr><td></td></tr>
</table>

<table>
<tr><td colspan="2">colspan=2</td></tr>
<tr><td></td><td></td></tr>
</table>

4.11 Formularios

Los formularios son utilizados comúnmente para realizar búsquedas o bien para introducir datos personales por ejemplo en sitios de comercio electrónico.

Usando HTML se puede únicamente enviar el formulario a un correo electrónico. Si se quiere procesar el formulario mediante un programa, se tendrán que emplear otros lenguajes más sofisticados.

Los formularios son definidos por medio de las etiquetas <*form*> y </*form*>. Entre estas dos etiquetas se colocarán todos los elementos que componen el formulario.

Atributos importantes para <form>:

`action` Define el tipo de acción a llevar acabo con el formulario (enviarlo a una dirección de correo electrónico o procesar los datos con un script).

method Especifica la forma en la que el formulario es enviado. Los dos valores posibles que puede tomar este atributo son *POST* y *GET*[≈].

Existe una gran diversidad de alternativas a la hora de crear los formularios. Los elementos más importantes de los formularios se describen a continuación.

4.11.1 Cajas de texto

Para colocar una caja de texto en el formulario, se utilizará la etiqueta <*input*>.

Atributos importantes de <input>

type Para que el elemento del formulario sea una caja de texto corto, este atributo debe tomar el valor *text.*
name Define el nombre de la caja de texto (este va a ser el nombre de la variable, si el formulario lo procesa un script).
size Define el tamaño de la caja de texto en número de caracteres.
maxlenght Indica el tamaño máximo del texto que puede ser introducido.
value Asigna un valor definido a la caja de texto.

Ejemplo:

```
<input type="text" name="fecha" value="dd/mm/aaaa">
```

4.11.2 Área de texto

Si se quiere un área de texto donde se pueda escribir cómodamente sobre un espacio compuesto de varias líneas, se deben utilizar las etiquetas <*textarea*> y *</textarea>.*

Atributos importantes para <textarea>

name Define el nombre del área de texto (este va a ser el nombre de la variable, si el formulario lo procesa un script).
rows Define el número de líneas del área de texto.
cols Define el número de columnas del campo de texto.

[≈] El método **GET** añade los argumentos del formulario al URL recogido en action (utilizando como separador de las variables la "?"). El método de uso más normal es **POST**, y en el caso de que estemos mandando el formulario a nuestra dirección de correo electrónico es obligado usarlo.

Ejemplo:

```
<textarea name="comentario" rows="10" cols="40">
     Escribe tu comentario aquí
</textarea>
```

Salida:

Escribe tu comentario aquí

4.11.3 Lista de opciones

Las listas desplegables permiten elegir una o varias de las múltiples opciones que proponen. Para construir una lista desplegable o lista de opciones se deben utilizar las etiquetas *<select> y </select>.*

Atributos importantes para <select>

name	Define el nombre de la lista desplegable.
size	Indica el número de valores mostrados de la lista.
multiple	Permite la selección de varios elementos de la lista.

Cada opción que se quiere incluir en la lista desplegable, se debe indicar con la etiqueta *<option>.*

Atributos importantes para <option>

selected	Este atributo simplemente indica la opción elegida por default.
value	Define el valor de la opción que será enviado al programa o correo electrónico si el usuario elige esa opción.

Ejemplo:

```
<select name="estacion" size="3" multiple>
<option value="1">Primavera</option>
<option value="2">Verano</option>
<option value="3">Otoño</option>
<option value="4">Invierno</option>
</select>
```

Salida:

Si el usuario elige primavera, lo que le llegara al programa (o correo) es una variable llamada *estacion* que tendrá como valor 1.

4.11.4 Botones de radio

Este elemento es de gran utilidad si se quiere obligar al usuario a elegir únicamente una de las opciones que se le proponen.

La etiqueta empleada en este caso es <*input*> en la cual se tendrá el atributo *type* ha de tomar el valor *radio*.

Ejemplo:

```
<input type="radio" name="estacion" value="1">Primavera    <br>
<input type="radio" name="estacion" value="2">Verano       <br>
<input type="radio" name="estacion" value="3">Otoño        <br>
<input type="radio" name="estacion" value="4">Invierno
```

Salida:

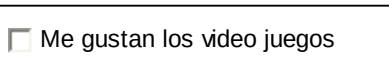
4.11.5 Cajas de validación

Este tipo de elementos pueden ser activados o desactivados por el usuario con un simple clic. Nuevamente se utilizará la etiqueta <*input*> pero el atributo *type* deberá tomar el valor *checkbox.*

Ejemplo:

```
<input type="checkbox" name="juegos">Me gustan los videojuegos
```

Salida:



La información que llegará al correo (o al programa) será del tipo:

```
juegos = on (u off dependiendo si ha sido activada o no)
```

4.11.6 Botón de envió

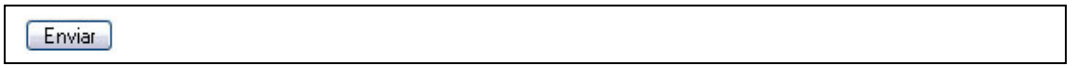
Permiten finalizar el proceso de llenado del formulario y hacerlo llegar a su gestor.

Para incluir un botón de envió se utiliza la etiqueta <*input*>, esta vez el valor que debe tomar el atributo *type* debe ser *submit*.

Ejemplo:

```
<input type="submit" value="Enviar">
```

Salida:



4.11.7 Botón de borrado

Este botón permitirá borrar el formulario por completo en el caso de que el usuario desee rehacerlo desde el principio.

Ejemplo:

```
<input type="reset" value="Borrar">
```

A diferencia del botón de envío, indispensable en cualquier formulario, el botón de borrado resulta meramente optativo y no es utilizado frecuentemente.

4.11.8 Datos ocultos

Algunas veces puede resultar práctico enviar datos definidos por nosotros mismos, que ayuden al script en su procesamiento del formulario. Estos datos no se muestran en la página, pero estas variables pueden ser utilizadas en el código fuente del script gestor.

Ejemplo:

```
<input type=hidden name="bandera" value="1">
```

5. Programación con PHP

Antes de comenzar con la programación con PHP, es necesario hablar sobre los servidores WWW. Un tema importante debido a que PHP es un lenguaje de lado servidor.

5.1 ¿Qué es un servidor WWW?

Un servidor WWW es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML.

Un servidor WWW se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP, que se conoce como navegador. El navegador realiza una petición al servidor y éste le responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. 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.

5.2 ¿Cómo funciona HTTP?

1. El *cliente* HTTP abre una conexión.
2. El *servidor* manda un “acknowledge” notificando que se ha abierto una sesión.
3. El *cliente* envía su “request message” solicitando un recurso.
4. El *servidor* responde con “response message” que contiene el recurso solicitado y cierra la conexión.

5.3 Servidores WWW Populares

A continuación se mencionan los servidores WWW más populares actualmente:

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

5.4 Criterios de Selección

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

En el diplomado se trabajó con Apache Web Server, a continuación se mencionan algunas de las características más importantes de Apache.

5.5 Características de Apache

- Soporta un gran número de transacciones.
- Configurable para diferentes entornos de trabajo.
- Alto nivel de seguridad.
- Disponible para una gran variedad de plataformas.
- Soporte para servicio de proxy.
- Soporte para granjas de servidores.
- Soporte para Scripting Languages integrados como módulos (por ejemplo PHP)
- Incluye el código fuente del servidor
- Soporte para accesos restringidos
- Soporte para SSL[*].
- Es gratuito.

5.6 Instalación de Apache

```
1. gzip -d httpd-2_0_NN.tar.gz      EXTRAER
2. tar xvf httpd-2_0_NN.tar
5. cd httpd-2_0_NN
6. ./configure --prefix=PREFIX      CONFIGURAR
7. make                             COMPILAR
8. make install                     INSTALAR
```

NN debe ser reemplazado por el número de la versión, y PREFIX debe ser reemplazado con la ruta donde se desee instalar, si no se especifica PREFIX por default se instala en /usr/local/apache2.

[*] SSL (Secure Sockets Layer) es un protocolo que proporciona comunicaciones seguras en Internet, permite cifrar la conexión, incluso garantiza la autentificación. Se basa en la criptografía asimétrica y en el concepto de los certificados.

5.7 Configuración del Servidor

Apache es administrado por más de 150 ***directivas*** las cuales permiten agregarle funcionalidad. En Linux el administrador controla qué directivas estarán disponibles de acuerdo a los módulos con los que se compila.

La configuración de Apache mediante directivas es clasificada en 3 grupos:

- Global Environment
- Main Server
- Virtual Servers

5.7.1 User

La directiva User especifica el identificador de usuario con el que el servidor responderá a las peticiones. Para usar esta directiva, el servidor debe haber sido iniciado como root. Si se inicia Apache con un usuario distinto de root, no se podrá cambiar a un usuario con menores privilegios, y el servidor continuará ejecutándose con el usuario original.

Sintaxis:	`User` *`unix-userid`*
Valor por defecto:	`User #-1`

5.7.2 Group

La directiva Group determina el grupo con el que el servidor atenderá las peticiones. Para usar esta directiva, el servidor debe haber sido iniciado con el usuario root. Si inicia el servidor con un usuario que no sea root, el servidor no podrá cambiarse al grupo especificado, en lugar de esto continuará ejecutándose con el grupo del usuario que lo inició.

Sintaxis:	`Group` *`unix-userid`*
Valor por defecto:	`Group #-1`

5.7.3 Port

Esta directiva define cual es el puerto en el que opera el servidor Web.

Sintaxis:	`Port` *`numero-puerto`*
Valor por defecto:	`Port #80`

5.7.4 ServerAdmin

ServerAdmin especifica la dirección de email que el servidor incluye en cualquier mensaje de error que envía al cliente.

Sintaxis:	**ServerAdmin** correo

5.7.5 DocumentRoot

Esta directiva define la ruta absoluta donde se almacenarán los archivos que se desean publicar.

Sintaxis:	**DocumentRoot** ruta
Valor por defecto:	**DocumentRoot** /usr/local/apache/htdocs

5.7.6 ServerRoot

La directiva ServerRoot especifica el directorio en el que ha sido instalado el servidor.

Sintaxis:	**ServerRoot** ruta
Valor por defecto:	**ServerRoot** /usr/local/apache

5.7.7 MinSpareServers

La directiva MinSpareServers fija el número mínimo de procesos hijo en espera. Un proceso en espera es aquel que no está atendiendo ninguna petición.

Sintaxis:	**MinSpareServers** número
Valor por defecto:	**MinSpareServers** 5

5.7.8 MaxSpareServers

La directiva MaxSpareServers determina el número máximo de procesos hijo en espera deseado. Si hay más de MaxSpareServers procesos hijo en espera, entonces el proceso padre elimina el exceso.

Sintaxis:	**MaxSpareServers** número
Valor por defecto:	**MaxSpareServers** 10

5.7.9 StartServers

La directiva StartServers especifica el número de procesos hijo que se crean al iniciar Apache. Como el número de procesos está controlado dinámicamente según la carga del servidor, no hay normalmente ninguna razón para modificar el valor de este parámetro.

Sintaxis:	**StartServers** number

5.7.10 MaxClients

La directiva MaxClients especifica el límite de peticiones simultáneas que serán atendidas. Cualquier intento de conexión por encima del límite MaxClients se pondrá en cola, hasta llegar a un límite basado en el valor de la directiva ListenBacklog.

Sintaxis:	**MaxClients** number

5.7.11 ErrorDocument

Es lo que el servidor devuelve al cliente si se produce algún error. Se puede configurar Apache para hacer una de las siguientes 4 cosas:

1. Devolver un mensaje de error estándar.
2. Devolver un mensaje de error personalizado.
3. Redireccionar la petición a una ruta-URL local.
4. Redireccionar la petición a una URL externa.

Sintaxis:	**ErrorDocument** código-error documento

5.7.12 PidFile

La directiva PidFile especifica el archivo en el que el servidor guarda el ID del proceso demonio de escucha (daemon).

Sintaxis:	**PidFile** *filename*
Valor por defecto:	**PidFile** logs/httpd.pid

5.7.13 HostnameLookups

Esta directiva activa la resolución de DNS de manera que los nombres de host puedan ser guardados en los archivos log (y pasados a CGIs/SSIs en REMOTE_HOST).

Sintaxis:	**HostnameLookups** On|Off|Double
Valor por defecto:	**HostnameLookups** Off

5.7.14 <Directory>

Las directivas <Directory> y </Directory> se usan para englobar un grupo de directivas que se aplicarán solamente al directorio especificado y a sus subdirectorios. Puede incluir a cualquier directiva cuyo uso esté permitido en un contexto <directory>.

Sintaxis:	**<Directory** *directorio***> ... </Directory>**

Ejemplo:

```
<Directory />
      AllowOverride None
</Directory>

<Directory /home/>
      AllowOverride FileInfo
</Directory>
```

5.7.15 Options

La directiva Options controla qué funcionalidades del servidor están disponibles en un directorio en particular.

En *option* puede especificar *None*, en cuyo caso ninguna funcionalidad adicional estará activada, o puede especificar una o más de las siguientes opciones:

`All`
Todas las opciones excepto MultiViews. Este es el valor por defecto.

`ExecCGI`
Se permite la ejecución de scripts CGI usando mod_cgi.

`Includes`
Permite el uso de Server-side includes, del módulo mod_include.

`Indexes`
Si se produce una petición a una URL que se corresponde con un directorio, y no hay DirectoryIndex (por ejemplo, index.html) en ese directorio, entonces mod_autoindex devolverá una lista con los contenidos del directorio.

`FollowSymLinks`
El servidor seguirá los enlaces simbólicos en este directorio

Sintaxis:	**`Options`** [+|-]*option* [[+|-]*option*] ...
Valor por defecto:	**`Options`** `All`

Ejemplo:

```
<Directory /web/docs>
      Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
      Options +Includes -Indexes
</Directory>
```

5.8 CGI (Common Gateway Interface)

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

Permiten generar páginas dinámicas y fueron las primeras alternativas para generar dinamismo en un sitio Web. Sus principales elementos de configuración son los siguientes:

5.8.1 ScriptAlias

Esta directiva convierte las solicitudes vía URL a la ruta absoluta donde residen los CGI's.

Sintaxis: `ScriptAlias URL-path file-path|directory-path`

Ejemplo:

```
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

5.8.2 AddHandler

Permite que determinada extensión sea relacionada a un evento en particular, en el caso de los CGI's lo que se indica es que la extensión .cgi queda identificada como un Script.

Sintaxis: `AddHandler handler-name extension [extension] ...`

Ejemplo:

```
AddHandler cgi-script .cgi
```

5.9 Manejo de sitios virtuales

Los sitios Web virtuales permiten tener varios Sitios Web independientes dentro de la misma máquina sin tener que lanzar varias instancias de Apache. De esta forma se pueden optimizar recursos.

Existen dos formas de definir sitios virtuales:

Por IP	Se identifica a cada servidor por una IP diferente.
Por nombre:	El servidor Web escucha peticiones en una única IP y es el nombre de la página el que define qué contenido se va a mostrar.

5.9.1 <VirtualHost>

<VirtualHost> y </VirtualHost> se usan para incluir un grupo de directivas que se aplicarán solo a un host virtual en particular. Cualquier directiva que esté permitido usar en un contexto virtual host puede usarse. Cuando el servidor recibe una petición de un documento de un host virtual en concreto, usa las directivas de configuración incluidas en la sección <VirtualHost>.

Sintaxis:
```
<VirtualHost addr[:port] [addr[:port]] ...> ...
</VirtualHost>
```

Ejemplo:

```
<VirtualHost 10.1.2.3>
      ServerAdmin webmaster@host.foo.com
      DocumentRoot /www/docs/host.foo.com
      ServerName host.foo.com
      ErrorLog logs/host.foo.com-error_log
      TransferLog logs/host.foo.com-access_log
</VirtualHost>
```

5.10 Introducción a PHP

PHP es el lenguaje de lado servidor más extendido en la Web. Este lenguaje ha tenido gran aceptación en la comunidad de webmasters debido principalmente a su potencia y simplicidad.

PHP permite introducir sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas en un lenguaje distinto de HTML.

Originalmente PHP fue concebido para entornos UNIX, pero actualmente es un lenguaje multiplataforma.

5.11 Tareas principales de PHP

- Funciones de correo electrónico.
- Gestión de bases de datos.
- Gestión de archivos.
- Tratamiento de imágenes.

5.12 Sintaxis de PHP

El código PHP se escribe junto con el código HTML, por eso se debe delimitar el código PHP con etiquetas.

Los modos de abrir y cerrar etiquetas son los siguientes:

```
<?          y     ?>
<?php       y     ?>
<script language="php"> </script>
```

5.13 Funcionamiento de una página PHP

El funcionamiento de una página PHP a grandes rasgos es el siguiente:

El servidor reconoce la extensión correspondiente a la página PHP (.html, .php, php4, etc.) y antes de enviarla al navegador va a encargarse de interpretar y ejecutar todo aquello que se encuentre entre las etiquetas correspondientes al lenguaje PHP. Por ultimo el código HTML restante lo enviará sin más, ya que es absolutamente comprensible por el navegador.

5.14 Variables en PHP

Para definir una variable en PHP, la sintaxis general es la siguiente:

```
$nombre-variable
```

5.14.1 Variables numéricas

Este tipo de variables almacenan cifras.

Ejemplo:

```
Enteros     $enteros=2005
Reales      $real=3.1416
```

5.14.2 Variables alfanuméricas

Almacenan cadenas de caracteres compuestas de números y/o cifras o texto.

Ejemplo:

```
$cadena=”HOLA MUNDO”
```

5.14.3 Arreglos (arrays)

Son colecciones de variables bajo un mismo nombre de variable, lo que las identifica es un índice.

```
$sentido[1]="ver";
$sentido[2]="tocar";
$sentido[3]="oir";
$sentido[4]="gusto";
$sentido[5]="oler";
```

5.15 Variables de sistema en PHP

Este tipo de variables informan sobre el servidor y sobre el cliente. La información de estas variables es atribuida por el servidor y no es posible modificar sus valores directamente mediante un script.

Estas son algunas de estas variables y la información que nos aportan:

Variable	Descripción
$HTTP_USER_AGENT	Informa principalmente sobre el sistema operativo, tipo y versión del navegador del cliente.
$HTTP_ACCEPT_LANGUAGE	Devuelve la o las abreviaciones de la lengua considerada como principal por el navegador.
$HTTP_REFERER	Indica la URL desde la cual el cliente tuvo acceso a la

	página.
$PHP_SELF	Regresa una cadena con la URL del script que está siendo ejecutado.
$HTTP_GET_VARS	Es un arreglo que almacena los nombres y contenidos de las variables enviadas al script por URL o por el método GET de los formularios.
$HTTP_POST_VARS	Es un arreglo que almacena los nombres y contenidos de las variables enviadas al script por el método POST de los formularios.
$HTTP_COOKIES_VARS	Es un arreglo que almacena los nombres y contenidos de las cookies.
$PHP_AUTH_USER	Almacena la variable *usuario* cuando se efectúa la entrada a páginas de acceso restringido.
$PHP_AUTH_PW	Almacena la variable *password* cuando se efectúa la entrada a páginas de acceso restringido.
$REMOTE_ADDR	Muestra la dirección IP del visitante.
$DOCUMENT_ROOT	Devuelve el *path* físico en el que se encuentra alojada la página en el servidor.

5.16 Variables superglobales

Estas variables hacen referencia a las mismas que se acceden por medio de los arrays del tipo *$HTTP_*_VARS*. A partir de PHP 5.0.0 se pueden desactivar con la directiva register_long_arrays.

Estas son algunas de estas variables y la información que aportan:

$GLOBALS	Contiene una referencia a cada variable disponible en el script.
$_SERVER	Son variables definidas por el servidor Web ó directamente relacionadas con el entorno en donde el script se esta ejecutando.
$_GET	Variables proporcionadas al script por medio de HTTP GET.
$_POST	Variables proporcionadas al script por medio de HTTP POST.
$_COOKIE	Variables proporcionadas al script por medio de HTTP cookies.
$_FILES	Variables proporcionadas al script por medio de la subida de

	archivos vía HTTP.
$_ENV	Variables proporcionadas al script por medio del entorno.
$_REQUEST	Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario y por lo tanto no se puede confiar en ellas.
$_SESSION	Variables registradas en la sesión del script.

5.17 Manejo de arreglos con PHP

Un arreglo es un conjunto de elementos bajo un mismo nombre de variable, cada elemento está catalogado por medio de una clave y deberá tener un valor.

Existen varias formas de declarar arreglos, las cuales se mencionan a continuación.

Sintaxis:

```
$nombre_arreglo[clave1] = valor1;
$nombre_arreglo[clave2] = valor2;
.
$nombre_arreglo[claven] = valorn;
```

Otra forma de definir un arreglo es la siguiente:

```
$nombre_arreglo = array("clave1"=> valor1,"clave2" => valor2" …);
```

Ejemplos:

```
$moneda["mexico"] = "Peso";
$moneda["francia"] = "Franco";
```

Otra forma de definir el mismo arreglo es:

```
$moneda = array("mexico"=> "Peso","francia" => Franco");
```

Es posible anidar arreglos, es decir, se puede tener un arreglo que tenga como elementos a otros arreglos.

Ejemplo:

```
<?    $pais=array
      (
            "mexico" =>array
            (
                  "nombre"=>"México",
                  "idioma"=>"Español",
                  "moneda"=>"Peso"
            ),
            "francia" =>array
            (
                  "nombre"=>"Francia",
                  "idioma"=>"Francés",
                  "moneda"=>"Franco"
            )
      );
      echo $pais["mexico"]["moneda"] //Saca en pantalla: "Peso"
?>
```

5.18 Funciones básicas para el manejo de arreglos

Existen muchas funciones para el manejo de arreglos, a continuación se describen las funciones básicas.

Función	Descripción
array_values (arreglo)	Lista los valores contenidos en *arreglo*.
asort(arreglo) y arsort(arreglo)	Ordena por orden alfabético directo o inverso en función de los valores.
count(arreglo)	Regresa el número de elementos de nuestro arreglo.
ksort(arreglo) y **krsort(arreglo)**	Ordena por orden alfabético directo o inverso en función de las claves.
list ($variable1, $variable2…) = arreglo	Asigna cada variable a cada uno de los valores del array.
next(arreglo), prev(arreglo), reset(arreglo) y **end(arreglo)**	Permiten moverse dentro del arreglo con un puntero hacia delante, atrás y al principio y al final.
each(arreglo)	Da el valor y la clave del elemento en el que se encuentra y mueve al puntero al siguiente elemento.
array_slice( )	Se utiliza cuando se quieren recortar algunos elementos del arreglo, sabiendo los índices de las casillas que se desean conservar. Recibe tres

	parámetros: el *arreglo*, el *índice del primer elemento* y el *número de elementos a tomar*, siendo este último parámetro opcional.
array_shift( )	Esta función extrae el primer elemento del arreglo y lo devuelve. Además, acorta la longitud del arreglo eliminando el elemento que estaba en la primera casilla. Solo recibe como parámetro el nombre del arreglo.
array_push( )	Inserta al final del arreglo una serie de elementos que se le indiquen por parámetro. El número de elementos del arreglo aumentará en función de los elementos que se le hayan pasado por parámetro.
array_merge( )	Esta función permite unir dos o más arreglos, regresa un arreglo con todos los elementos de los arreglos pasados.

Ejemplo:

```
<?    $animales = array("Lagartija", "Araña", "Perro");
      $numeros = array(12,34,56);

      //aumentamos el tamaño del array
      $resultado = array_merge($animales, $numeros);

      foreach ($resultado as $actual)
            echo $actual . "<br>";
?>
```

Salida:

```
Lagartija
Araña
Perro
12
34
56
```

5.19 Cadenas

Para asignar a una variable un contenido de tipo cadena, se debe escribir entre comillas.

Sintaxis: `$cadena = "Esta es la información de mi variable"`

Si se quiere ver en pantalla el valor de una variable o bien un mensaje cualquiera se utiliza la instrucción *echo*.

Ejemplo:

```
echo $cadena //sacaría "Esta es la información de mi variable"
```

5.19.1 Concatenación de cadenas

Para concatenar varias cadenas, simplemente se debe poner un "." entre ellas.

Ejemplo:

```
<?
      $cadena1 = "HOLA";
      $cadena2 = " MUNDO";
      $cadena3 = $cadena1.$cadena2;
      echo $cadena3 //El resultado es: "HOLA MUNDO"
?>
```

5.19.2 Funciones básicas para el manejo de cadenas

A continuación se describen algunas funciones básicas para el manejo de cadenas.

Función	Descripción
strcmp(cadena1,cadena2)	Esta función compara cadenas. Devuelve < 0 si *cadena1* es menor que *cadena2*; > 0 si *cadena1* es mayor que *cadena2* y 0 si son iguales.
strlen(cadena)	Esta función regresa la longitud de la cadena indicada.
strtolower(cadena)	Pasa a minúsculas una cadena.
strtoupper(cadena)	Devuelve la *cadena* con todas sus letras en mayúsculas.
substr(cadena, comienzo, longitud)	Devuelve la porción de *cadena* especificada por los parámetros *comienzo* y *longitud*.
trim(cadena)	Elimina espacios en blanco (u otros caracteres) del principio y final de una cadena.

5.20 Funciones

Una función es un conjunto de instrucciones que explotan ciertas variables y realiza una tarea específica.

Es posible crear nuestras propias funciones de la siguiente forma:

Sintaxis:

```
function nombre_funcion ($arg_1, $arg_2, ..., $arg_n)
{
      Bloque de instrucciones;
      //Valor que regresa la función
      return $retval;
}
```

Ejemplo:

```
<html>
<head>
    <title> Función 1</title>
</head>
<body>
<?
      function escribe_separado($cadena)
      {
            for ($i = 0; $i < strlen($cadena); $i++)
            {
                  echo $cadena[$i];
                  if ($i < strlen($cadena)-1)
                        echo "-";
            }
      }
      //Llamada a la función escribe_separado
      escribe_separado("hola");
      echo "<p>";
      escribe_separado ("Texto más largo, a ver lo que hace");
?>
</body>
</html>
```

Descripción:

Esta función imprimirá cada caracter de la cadena que se le pase como parámetro separado por un “-“.

Salida:

```
h-o-l-a

T-e-x-t-o- -m-á-s- -l-a-r-g-o-,- -a- -v-e-r- -l-o- -q-u-e- -h-a-c-e
```

PHP permite pasar parámetros a una función ya sea “por valor” o “por referencia”[*].

5.21 Operadores

Los operadores permiten crear, modificar y comparar variables dentro de un programa. A continuación se describen los diferentes tipos de operadores que se pueden utilizar en PHP.

5.21.1 Operadores aritméticos

Permiten realizar operaciones numéricas con nuestras variables:

+	Suma
-	Resta
*****	Multiplicación
/	División
%	Módulo (Devuelve el residuo de la división)

5.21.2 Operadores de comparación

Se utilizan para comparar dos variables y verificar si cumple o no la propiedad del operador.

==	Igualdad
!=	Desigualdad
<	Menor que
>	Mayor que
<=	Menor igual que
>=	Mayor igual que

5.21.3 Operadores lógicos

Se usan en combinación con los operadores de comparación cuando la expresión de la condición lo requiere.

&&	AND
||	OR
!	NOT

[*] Cuando se pasa **“por valor”** un parámetro a una función, el cambio del valor de un parámetro dentro de una función no afecta al valor de la variable original.
En este caso el cambio del valor de un parámetro dentro de una función sí afecta al valor de la variable original.

5.21.4 Operadores de incremento

Sirven para aumentar o disminuir de una unidad el valor de una variable

++$variable	Aumenta en 1 el valor de $variable
--$variable	Reduce en 1 el valor de $variable

5.21.5 Operadores combinados

Una forma habitual de modificar el valor de las variables es mediante los operadores combinados:

$variable += 10	Suma 10 a $variable
$variable -= 10	Resta 10 a $variable
$variable.= "añado"	Concatena las cadenas $variable y "añado"

5.22 if-else

En esta estructura de control primero se evalúa la condición, si la condición se cumple se ejecuta el *bloque de instrucciones 1,* si no, se ejecutará el *bloque de instrucciones 2.*

Sintaxis:

```
if (condición)
{
      Bloque de instrucciones 1;
}
else
{
      Bloque de instrucciones 2;
}
```

Ejemplo:

```
<?
      $a=1;
      $b=2;

      if($a > $b)
            echo "a es mayor que b";
      else
            echo "a NO es mayor que b";
?>
```

Salida:

```
a NO es mayor que b
```

5.23 while

Esta estructura permite repetir instrucciones mientras se cumpla una condición.

Sintaxis:

```
while (condición)
{
      Bloque de instrucciones a repetir;
}
```

Ejemplo:

```
<?
      function busca_caracter($cadena, $caracter)
      {
            $i = 0;
            //Recorremos la cadena en busca del caracter
            while ($i < strlen($cadena) && $cadena[$i] != $caracter)
                  $i++;
            //Si recorrió toda la cadena no está el caracter
            if($i == strlen($cadena) )
                  echo "La $caracter NO está en \"$cadena\" ";
            else
                  echo "La $caracter YA está en \"$cadena\" ";
      }
      busca_caracter("hola mundo", 'o');
      echo "<br>";
      busca_caracter("hola mundo", 'x');
?>
```

Descripción:

Esta función que recibe 2 parámetros, el primer parámetro será una cadena, y el segundo un carácter a buscar en la cadena, la función imprimirá un mensaje diciendo si está o no el carácter en la cadena.

Salida:

```
La o YA está en "hola mundo"
La x NO está en "hola mundo"
```

5.24 for

La estructura *for* funciona de la siguiente manera:

La primera expresión *expr1* se evalúa incondicionalmente una vez al principio del ciclo.

Al comienzo de cada iteración, se evalúa *expr2*. Si se cumple la condición, el ciclo continúa y se ejecuta el *bloque de instrucciones*. Si no se cumple, la ejecución del ciclo finaliza.

Al final de cada iteración, se evalúa *expr3*.

Sintaxis:

```
for (expr1; expr2; expr3)
{
        bloque de instrucciones;
}
```

Ejemplo:

```
<?      //Bucle for sencillo
        for ($i=1; $i <=5; $i++)
                echo "$i ";

        echo "<p> For anidado</p>";
        //Bucle for anidado
        for ($j=1; $j<=3; $j++)
        {
                for ($i=1; $i <=5; $i++)
                        echo "$i ";

                echo "<br>";
        }
?>
```

Descripción:

El primer ciclo *for* simplemente sirve para imprimir una lista del 1 al 5. En la segunda parte del código utilizamos 2 ciclos *for*, el bucle interno sirve para imprimir una lista del 1 al 5, mientras que el bucle for externo sirve para imprimir 3 veces las listas del 1 al 5.

Salida:

```
1 2 3 4 5

For anidado

1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

5.25 foreach

Esta estructura repetitiva permite recorrer fácilmente los valores de un arreglo.

Sintaxis:

```
foreach ($arreglo as $clave=>$valor)
{
      bloque de instrucciones;
}
```

Ejemplo:

```
<?    $paises = array("mx" => "México","fr" => "Francia",
                      "ca" => "Canada",);

      foreach ($paises as $clave=>$valor)
      {
            echo "Clave: $clave País: $valor <br>";
      }
?>
```

Descripción:

Este script simplemente mostrará en pantalla el contenido del arreglo *paises*.

Salida:

```
Clave: mx País: México
Clave: fr País: Francia
Clave: ca País: Canada
```

5.26 Paso de variables por la URL

Para pasar las variables de una página a otra se puede hacer introduciendo dicha variable dentro del enlace hipertexto de la página destino.

Sintaxis:

```
<a href="destino.php?variable1=valor1&variable2=valor2&...">
      Mi enlace
</a>
```

Ejemplo:

Archivo origen.html

```
<html>
<head>
<title>origen.html</title>
</head>
<body>
<a href="destino.php?saludo=hola&texto=esto es una variable
texto">paso variables saludo y texto a la página destino.php</a>
</body>
</html>
```

Archivo destino.php

```
<html>
<head><title>destino.php</title></head>
<body>
<?    echo "variable \$saludo: $saludo <br>\n";
      echo "variable \$texto: $texto <br>\n"
?>
</body>
</html>
```

Descripción:

Se tienen dos páginas, *origen.html* y *destino.php.* La primera pagina contiene una liga al archivo *destino.php,* esta le pasará dos variables por la URL (*saludo* y *texto*). Finalmente el archivo *destino.php* desplegará el contenido de las variables.

Salida:

Archivo origen.html

```
Paso variables saludo y texto a la página destino.php
```

Archivo origen.php

```
Variable $saludo: hola
Variable $texto: Esto es una variable texto
```

5.27 Procesar variables de Formularios

Si se quieren procesar las variables provenientes de un formulario, PHP permite hacerlo de una manera sencilla. Primero se presentará en una página el formulario clásico a llenar y estas variables las procesa un segundo archivo.

Ejemplo:

Archivo formulario.html

```
<html>
<head><title>formulario.html</title></head>
<body>
   <form method="post" action="destino2.php">
      Nombre<br>
      <input type="text" name="nombre"><br>
      Apellidos <br>
      <input type="text" name="apellidos"><br>
      <input type="submit" value="Enviar consulta">
    </form>
</body></html>
```

Archivo destino.php

```
<html>
<head><title>destino2.php</title></head>
<body>
<?
      echo "variable \$nombre: $nombre <br>\n";
      echo "variable \$apellidos: $apellidos <br>\n"
?>
</body></html>
```

Descripción:

El formulario tendrá dos cajas de texto, *nombre* y *apellidos,* que también serán los nombres de las variables a procesar en el script *destino.php.* El script *destino.php* simplemente imprimirá el contenido de estas dos variables.

Salida:

Archivo formulario.html

Archivo destino.php

```
Variable $nombre: Edgar
Variable $apellidos: Peña
```

5.28 Sesiones

Una sesión resulta muy práctica en los casos en los que se quiere conservar una variable en varios scripts diferentes y distantes unos de otros.

Características de las variables de sesión:

- Estas variables residen en el servidor
- Son específicas de un solo usuario definido por un identificador.
- Pueden ser utilizadas en la globalidad de nuestras páginas.

Para iniciar una sesión se puede hacer de dos formas distintas:

1. Declarar la apertura de sesión por medio de la función **`session_start()`**. Esta función crea una nueva sesión para un nuevo visitante o bien recupera la que está siendo llevada a cabo.

2. Declarar una variable de sesión por medio de la función *session_register('variable')*. Esta función, además de crear o recuperar la sesión para la página en la que se incluye también sirve para introducir una nueva variable de tipo sesión.

Las sesiones deben ser iniciadas al principio del script. Antes de abrir cualquier etiqueta o de imprimir cualquier cosa. En caso contrario recibiremos un error.

Ejemplo:

```
<?  session_register('contador');  ?>
<html>
<head><title>contador.php</title></head>
<body>
<?
      if(isset($contador)==0)
            $contador=0;
      ++$contador;
      echo "<a href=\"contador.php\">Has recargado esta página
            $contador veces</a>";
?>
</body>
</html>
```

Descripción:

Este es un ejemplo clásico de un contador, este contador deberá aumentar en 1 cada vez que se recarga la página o damos un clic en el enlace.

La condición *if* verifica si la variable *$contador* ha sido inicializada. La función *isset* se encarga de dar un valor cero cuando una variable no ha sido inicializada.

Salida:

Has recargado esta página 5 veces

5.28.1 Funciones importantes para la gestión de sesiones

session_id( )	Nos devuelve el identificador de la sesión.
session_destroy( )	Da por abandonada la sesión eliminando variables e identificador.
session_unregister('variable')	Abandona una variable sesión.

6. PHP con MYSQL

Uno de los puntos fuertes de PHP es la posibilidad de explotar *bases de datos*[*] usando funciones simples y potentes. Las bases de datos permiten almacenar contenidos de una forma sistemática para clasificarlos, buscarlos y editarlos rápida y fácilmente.

MySQL es uno de los Sistemas Gestores de bases de Datos (SQL) más populares desarrolladas bajo la filosofía de código abierto.

6.1 Instalación de MYSQL 4.0.26

```
1.   Agregar el grupo mysql al sistema
     groupadd mysql

2.   Agregar el usuario mysql al grupo mysql
     useradd -g mysql mysql

3.   Descomprimir el archivo mysql-4.0.26.tar.gz
     tar -zxvf mysql-4.0.26.tar.gz

4.   Cambiamos al directorio mysql-4.0.26
     cd mysql-4.0.26/

5.   Configurar mysql en /usr/local/mysql
     ./configure --prefix=/usr/local/mysql

6.   make
7.   make install

8.   Copiar archivo de configuración
     cp support-files/my-medium.cnf /etc/my.cnf

9.   Cambiamos de directorio a /usr/local/mysql/
     cd /usr/local/mysql/

10.  Instalar la base de datos
     bin/mysql_install_db --user=mysql

11.  Cambiar de dueño (a root) el directorio mysql
     chown -R root .

12   Cambiar de dueño (a mysql) el directorio mysql/var
     chown -R mysql var/

13.  Cambiar de grupo (a mysql) el directorio /mysql
     chgrp -R mysql .

14.  Levantar el servicio
     bin/mysqld_safe --user=mysql &
```

[*] Una **base de datos** es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior.

6.2 Introducción a SQL

El SQL (Structured Query Language), es un lenguaje estándar de comunicación de bases de datos. Es un lenguaje normalizado que nos permite trabajar con cualquier tipo de lenguaje (ASP, PHP, etc.) en combinación con cualquier tipo de bases de datos (Oracle, Postgresql, Mysql, Sybase, etc.).

6.2.1 Tipos de datos comunes

Cada manejador de bases de datos tiene sus propios tipos de datos, pero existe un conjunto de tipos de datos comunes, que están representados en la totalidad de estos manejadores. Estos tipos de datos comunes son los siguientes:

Tipo	Descripción
alfanuméricos	Contienen cifras y letras. Presentan una longitud limitada (255 caracteres).
numéricos	Existen de varios tipos, principalmente, enteros (sin decimales) y reales (con decimales).
booleanos	Simplemente almacenan: Verdadero o falso (Sí o No).
fechas	Almacenan fechas facilitando posteriormente su explotación.
memos	Son campos alfanuméricos de longitud ilimitada.
autoincrementables	Son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado.

6.2.2 Insertar un nuevo registro

Sintaxis:

```
insert into nombre_tabla (nombre_campo1, nombre_campo2,...)
values (valor campo1, valor campo2...)
```

Ejemplo:

```
insert into empleados (nombre, apellidos, edad)
values ('edgar','peña',23)
```

Los campos no numéricos o booleanos van delimitados por comillas simples (‘).

6.2.3 Eliminar un registro

Sintaxis: `delete from nombre-tabla where condición`

Ejemplo:

```
delete from clientes where nombre='Juan'
```

Descripción:

Este ejemplo borrará todos los registros de los clientes que se llamen “Juan”. Hay que tener cuidado con esta instrucción ya que si no se especifica una condición con *where*, lo que se está haciendo es borrar toda la tabla.

6.2.4 Actualizar un registro

Sintaxis:

```
update      nombre_tabla
set         nombre_campo1=valor_campo1,
            nombre_campo2=valor_campo2,...
where       condiciones_de_selección
```

Ejemplo:

```
update clientes set nombre='José' where nombre='Pepe'
```

Descripción:

En este ejemplo se cambiará el nombre Pepe por el de José en todos los registros cuyo nombre sea Pepe.

6.2.5 Crear una tabla

La sintaxis para crear una tabla puede variar ligeramente de un manejador de bases de datos a otro ya que los tipos de campo aceptados no están completamente estandarizados.

Sintaxis:

```
create table nombre_tabla
(
      nombre_campo_1 tipo_1
      nombre_campo_2 tipo_2
      nombre_campo_n tipo_n
      key(campo_x,...)
)
```

Ejemplo:

```
create table pedidos
(
      pedido_id int(4) not null,
      cli_id int(4) not null,
      art_id int(4)not null,
      pedido_fecha date,
      pedido_cantidad int(4),
      pedido_total int(4), key(pedido_id,cli_id,art_id)
)
```

Descripción:

El nombre de la tabla será *pedidos*, los campos *pedido_id,cli_id* y *art_id* serán de tipo entero, el campo *pedido_fecha* será de tipo fecha, los campos *pedido_cantidad* y *pedido total* serán de tipo entero. Finalmente se especifican los campos que conformarán la llave primaria de la tabla con la palabra reservada *reí*.

Si un campo se define como *not null,* quiere decir que forzosamente deberá tener un valor.

6.2.6 Seleccionar algunos campos

Se puede realizar la selección parcial de una tabla mediante la instrucción *select.* La sintaxis básica se menciona a continuación:

Sintaxis: `select campo(s) from tabla`

Ejemplo:

```
select nombre, apellidos from clientes
```

6.2.7 Seleccionar todos los campos de la tabla

Para seleccionar todos los campos de una tabla se debe utilizar un "*" en lugar de especificar los nombres de los campos.

Sintaxis: `select * from nombre-tabla`

Ejemplo:

```
select * from personas
```

6.2.8 Distinct

La palabra clave *distinct* se utiliza para regresar solo valores diferentes.

Sintaxis: `select distinct campo(s)from tabla`

Ejemplo:

```
select distinct compania from ordenes
```

Descripción:

Esta instrucción seleccionará todos los valores diferentes del campo *compania* de la tabla ordenes.

6.2.9 Order by

Se puede ordenar los resultados en función de uno o varios de sus campos.

Sintaxis: ***select*** campo(s) ***from*** tabla ***order by*** campo

Ejemplo:

```
select * from clientes order by pedidos desc
```

6.2.10 Operadores

Los operadores se utilizan después de la cláusula *where* y pueden ser combinados para optimizar nuestra selección.

Operadores matemáticos:

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto
=	Igual

Operadores lógicos

and
or
not

Otros operadores

like	Selecciona los registros cuyo valor de campo se asemeje, no teniendo en cuenta mayúsculas y minúsculas.
in,not in	Da un conjunto de valores para un campo para los cuales la condición de selección es (o no) valida.
is null	Selecciona aquellos registros donde el campo especificado está vacío.
between,and	Selecciona los registros comprendidos en un intervalo.
distinct	Selecciona los registros no coincidentes.
desc	Clasifica los registros por orden inverso.

Comodines

*	Sustituye a todos los campos.
%	Sustituye a cualquier cosa o nada dentro de una cadena.
_	Sustituye un solo carácter dentro de una cadena.

6.3 Tipos de datos en Mysql

Es importante determinar el tipo de dato que deberá tener cada campo de nuestra tabla, para ajustar el diseño de la base de datos y conseguir un almacenamiento óptimo con la menor utilización de espacio.

Los tipos de datos más importantes en Mysql son los siguientes:

Tipos numéricos

tinyint	Es un número entero con o sin signo.
bit o bool	Un número entero que puede ser 1 o 0.
smallint	Número entero con o sin signo.
mediumint	Número entero con o sin signo.
integer,int	Número entero con o sin signo.
bigint	Número entero con o sin signo.
float	Número pequeño en coma flotante de precisión simple.
real,double	Número en coma flotante de precisión doble.

Tipos fecha

date	Tipo fecha, almacena una fecha.
datetime	Combinación de fecha y hora.
timestamp	Combinación de fecha y hora.
time	Almacena una hora.
year	Almacena un año.

Tipos de cadena

char(n)	Almacena una cadena de longitud fija.
varchar(n)	Almacena una cadena de longitud variable.

Tipos blob*

tinytext	Columna con una longitud máxima de 255 caracteres.
tinyblob	Columna con una longitud máxima de 255 caracteres.
blob	Un texto con un máximo de 65535 caracteres.
text	Un texto con un máximo de 65535 caracteres.
mediumblob	Un texto con un máximo de 16.777.215 caracteres.
mediumtext	Un texto con un máximo de 16.777.215 caracteres.
longblob	Un texto con un máximo de caracteres 4.294.967.295.
longtext	Un texto con un máximo de caracteres 4.294.967.295.
enum	Campo que puede tener un único valor de una lista que se especifica.

* Los tipos de dato **blob** se utilizan para almacenar datos binarios como pueden ser archivos.

6.4 Comandos básicos de Mysql

A continuación se mencionan los comandos básicos para trabajar con Mysql.

Arranque Mysql

Sintaxis: ***mysqld_safe*** **&**

Vista de las bases de datos existentes

Sintaxis: ***mysqlshow***

Vista de tablas dentro de una base de datos

Sintaxis: ***mysqlshow*** basedatos

Vista de estructura de una tabla

Sintaxis: ***mysqlshow*** basedatos tabla

Dar de baja el servidor

Sintaxis: ***mysqladmin*** shutdown

Verificar estado del servidor Mysql

Sintaxis: ***mysqladmin*** version
mysqladmin status

Crear una base de datos desde el prompt

Sintaxis: ***mysqladmin create*** *basedatos*

Eliminar una base de datos desde el prompt

Sintaxis: ***mysqladmin drop*** basedatos

Respaldar una base de datos

Sintaxis: ***mysqldump*** basedatos > respaldo.sql

Recuperar información de un respaldo

Sintaxis: ***mysql*** respaldo < respaldo.sql

7. Introducción a la seguridad en cómputo

La seguridad puede entenderse como un conjunto de políticas y mecanismos que permiten garantizar la confidencialidad, la integridad y la disponibilidad de los recursos de un sistema. En la actualidad uno de los activos más importantes en una organización es la información.

En un principio las redes fueron diseñadas para el intercambio de información y recursos, en aquel entonces la seguridad no era un factor tomado en cuenta en el diseño de las redes, pues no se tenía una idea de las dimensiones que éstas llegarían a tener en el futuro.

Desafortunadamente el gran crecimiento de las redes ha incluido a individuos deshonestos que se introducen a los sistemas de información.

7.1 Algunos conceptos de seguridad

Es importante aclarar algunos conceptos importantes, para tener una visión generalizada de lo que es la seguridad informática.

Amenaza: Circunstancia o evento que puede causar daño violando la confidencialidad, integridad o disponibilidad.

Vulnerabilidad: Condición que tiene potencial para permitir que ocurra una amenaza, con mayor frecuencia e impacto. En el argot de la seguridad computacional una vulnerabilidad también es conocida como un *hoyo.*

Riesgo: Es el potencial para pérdida o falla de un sistema, como respuesta a las siguientes preguntas:

- ¿Qué podrá pasar (o cuál es la amenaza)?
- ¿Qué tan malo puede ser (impacto o consecuencia)?
- ¿Qué tan frecuente puede ocurrir?
- ¿Qué tanta certidumbre se tiene en las primeras 3 respuestas (grado de confianza)?

Exploit: Se refiere a la forma de explotar una vulnerabilidad. Es un término muy enfocado a herramientas de ataque, sobre equipos de cómputo.

Ataques: Cualquier acción que cause la destrucción, modificación, o retraso del servicio no autorizado.

Confidencialidad: Un sistema posee la propiedad de confidencialidad si, la información manipulada por éste no es disponible ni puesta en descubierto para usuarios, entidades o procesos no autorizados. 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.

Autenticación: 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:

- Autenticación del proveedor de bienes o servicios
- Autentificación del cliente

Disponibilidad: La disponibilidad es la garantía de que los usuarios autorizados puedan acceder a la información y recursos cuando los necesiten.

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.

7.2 Estructura de la seguridad informática

Estrategias y políticas

Estrategias de administración para seguridad informática y políticas, estándares, guías o directivas usadas para comunicar estas estrategias a la organización.

Administración de la organización

Procesos que se dirigen hacia políticas profesionales y programas de capacitación, administración de cambios y control, administración de seguridad y otras actividades necesarias.

Monitorización de eventos

Procesos reactivos que permite a la administración medir correctamente la implementación de políticas e identificar en que momento las políticas necesitan cambios.

Tecnología informática

Es la tecnología necesaria para proveer la apropiada protección y soporte en los distintos procesos involucrados en la organización. La seguridad informática abarca un amplio rango de estrategias y soluciones, tales como:

- *Control de acceso:* Básicamente, el papel del control de acceso es identificar a la persona que desea acceder al sistema y a sus datos, y verificar la identidad de dicha persona.

Planificación y administración del sistema

Planificación, organización y administración de los servicios relacionados con la informática, así como políticas y procedimientos para garantizar la seguridad de los recursos de la organización.

Cifrado

La encriptación y la desencriptación de la información manipulada, de manera tal que sólo las personas autorizadas pueden acceder a ella.

Seguridad de la red y de comunicaciones

Controlar problemas de seguridad a través de las redes y los sistemas de telecomunicaciones.

Seguridad física

Otro aspecto importante de la seguridad informática es la seguridad física de sus servicios, equipos informáticos y medios de datos reales; para evitar problemas que pueden tener como resultado: Pérdida de la productividad, pérdida de ventaja competitiva y sabotajes intencionados. Algunos de los métodos de prevenir el acceso ilegal a los servicios informáticos incluyen:

- Claves y contraseñas para permitir el acceso a los equipos.
- Uso de cerrojos y llaves.
- Fichas ó tarjetas inteligentes.
- Dispositivos biométricos (Identificación de huellas dactilares, lectores de huellas de manos, patrones de voz, firma / escritura digital, análisis de pulsaciones y escáner de retina, entre otros).

7.3 Control de Acceso

Existen dos modelos para la autenticación: DAC y MAC

- **Control de acceso discrecional (DAC):** Un usuario bien identificado (típicamente, el creador o propietario del recurso) decide cómo protegerlo estableciendo cómo compartirlo, mediante controles de acceso impuestos por el sistema.

- **Control acceso mandatario (MAC):** Es el sistema quién protege los recursos, todo recurso del sistema y todo usuario tienen una etiqueta de seguridad.

Nombres de los usuarios

Cada persona que va a tener acceso a un sistema necesita un nombre de usuario para poder autenticarse ante él.

Grupos de usuarios

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

Super usuarios

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.

7.4 Administración básica de la seguridad

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 la 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.
- 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.

7.5 Monitoreo de Sistemas

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

Se deben monitorear los procesos que están ejecutándose de preferencia con scripts externos, es la mejor manera de ver el estado del sistema.

Es recomendable 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 del sistema.

Para facilitar la tarea, Linux pone a nuestra 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
```

Nos 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.

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.

7.6 Procesos

Un proceso es un concepto manejado por el sistema operativo, que consiste en el conjunto formado por:

- Las instrucciones de un programa destinadas a ser ejecutadas por el microprocesador.
- Su estado de ejecución en un momento dado, esto es, los valores de los registros del CPU para dicho programa.
- Su memoria de trabajo, es decir, la memoria que ha reservado y sus contenidos.

Un programa puede dar lugar a varios procesos. Un proceso puede estar detenido pero a diferencia de un programa existe una información de estado asociada al proceso.

PID y PPID

A cada proceso le corresponderá un número PID que le identifica totalmente. Es decir en un mismo momento es imposible que existan dos procesos con el mismo PID. Lo mismo que todos los procesos tienen un atributo PID que es el número de proceso que lo identifica en el sistema, también existe un atributo llamado PPID. Este número se corresponde con el número PID del proceso padre. Todos los procesos deben de tener un proceso que figure como padre.

7.7 Seguridad en redes

Las principales amenazas o riesgos que existen al utilizar las redes son los siguientes:

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.

Acceso no autorizado a computadoras y redes de computadoras: el acceso no autorizado a computadoras o 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, entre otras.

Ejecución de Programas que Modifican y Destruyen los Datos: las computadoras funcionan con programas, pero 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 de magnitud considerable.

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, proveedores de servicios o usuarios.

7.8 Detección de intrusos.

Los sistemas computarizados y aplicaciones están en permanente evolución, por tal razón pueden surgir nuevos puntos vulnerables. A pesar de los avances en los sistemas de seguridad, los usuarios no autorizados con herramientas muy sofisticadas tienen grandes posibilidades de acceder a las redes, sistemas o sitios de las organizaciones e interrumpir sus operaciones.

Factores que Propician el Acceso de Intrusos a la Redes y Sistemas

- Los sistemas operativos y las aplicaciones nunca estarán protegidos. Incluso si se protege el sistema nuevas vulnerabilidades aparecerán en el entorno todos los días, como las que actualmente representan los teléfonos, equipos inalámbricos y dispositivos de red.
- Falta de seguridad física en algunas empresas y falta de políticas de seguridad informática
- Los empleados no siempre siguen y reconocen la importancia de las políticas de seguridad.

- Requerimientos cada vez mayores de disponibilidad de redes y acceso a ellas.

Medidas para Controlar el Acceso de Intrusos

- Utilizar un firewall, que no es más que un dispositivo localizado entre la computadora anfitriona y una red, con el objeto de bloquear el tráfico no deseado de la red mientras permite el cruce de otro tráfico.
- Utilización y actualización de antivirus.
- Actualizar todos los sistemas, servidores y aplicaciones, ya que los intrusos por lo general aprovechan agujeros conocidos de seguridad.
- Desactivar los servicios innecesarios de redes.
- Eliminar todos los programas innecesarios.
- Analizar la red en busca de servicios comunes de acceso furtivo y utilizar sistemas de detección de intrusos los cuales permiten detectar ataques que pasan inadvertidos a un firewall y avisar antes o justo después de que se produzcan.
- Establecer la práctica de crear respaldos o backups.

7.9 Firewalls

Un firewall es un elemento de hardware o software utilizado en una red, para prevenir algunos tipos de comunicaciones prohibidas por las políticas de red, las cuales se fundamentan en las necesidades del usuario.

La configuración correcta de un firewall se basa en conocimientos considerables de los protocolos de red y de la seguridad de la computadora. Un pequeño error de configuración puede dejar a un firewall sin valor como herramienta de seguridad.

Tipos de firewalls

- Firewalls de capa de red. Funciona al nivel de la red de la pila de protocolos (TCP/IP) como filtro de paquetes IP o bien a nivel 2, de enlace de datos, no permitiendo que estos pasen el firewall a menos que se atengan a las reglas definidas por el administrador del cortafuegos o aplicadas por defecto como en algunos sistemas inflexibles de firewall.

- Cortafuegos de capa de aplicación. Trabaja en el nivel de aplicación. Analizan todo el tráfico de HTTP, (u otro protocolo), puede interceptar todos los paquetes que llegan o salen desde y hacia las aplicaciones que corren en la red. Este tipo de cortafuegos usa ese conocimiento sobre la información transferida para proveer un bloqueo más selectivo y para permitir que ciertas aplicaciones autorizadas funcionen adecuadamente. A menudo tienen la capacidad de modificar la información transferida sobre la marcha, a modo de engañar a las aplicaciones y hacerles creer que el firewall no existe.

Ventajas

- Protege de intrusiones. Solamente entran a la red las personas autorizadas basadas en la política de la red en base a las configuraciones.
- Optimización de acceso. Identifica los elementos de la red internos y optimiza que la comunicación entre ellos sea más directa. Esto ayuda a reconfigurar los parámetros de seguridad.
- Protección de información privada. Permite el acceso solamente a quien tenga privilegios a la información de cierta área o sector de la red.
- Protección contra virus. Evita que la red se vea infestada por nuevos virus que sean liberados.

8. PHP con PostgreSQL

PostgreSQL es el gestor de bases de datos de código abierto más avanzado actualmente, ofreciendo control de concurrencia multi-versión, soportando SQL, además de una gran variedad de lenguajes de programación.

Fue desarrollado originalmente en el Departamento de Ciencias de la Computación de U.C. Berkeley, fue pionero en muchos de los conceptos de objetos y relacionales que ahora están apareciendo en algunas bases de datos comerciales. Provee soporte para lenguajes SQL92/SQL99, transacciones, integridad referencial, procedimientos almacenados y extensibilidad de tipos.

8.1 Instalación de PostgreSQL

Si se desea instalar la versión más actual de PostgreSQL se deben seguir los siguientes pasos:

```
1. Configuración
./configure

2. Construcción
gmake

3. Asegurar el administrador
su

4. Prueba de regresión
gmake check

5. Instalando los archivos
gmake install

6. Crear una cuenta para postgresql
adduser postgresql

7. Crear la base de datos de instalación
mkdir /usr/local/pgsql/data
chown postgres /usr/local/pgsql/data
su – postgres
/usr/local/pgsql/bin/initdb –D /usr/local/pgsql/data

8. Arranque del servidor
/usr/local/pgsql/bin/postmaster –i –p 5432 –D /usr/local/pgsql/data &

9. Crear la base de datos
/usr/local/pgsql/bin/createdb test
/usr/local/pgsql/bin/psql test
```

8.2 Tipos de datos

PostgreSQL tiene un gran conjunto de datos nativos para los usuarios. Los usuarios también pueden agregar algunos tipos usando el comando DEFINE TYPE. A continuación se mencionan los tipos de datos más importantes en PosgreSQL.

Tipos numéricos

Tipo Numérico	Tamaño	Rango
int2	2 bytes	-32768 a +32767
int4	4 bytes	-2147483648 a +2147483647
int8	8 bytes	-9223372036854775808 a 9223372036854775807
float4	4 bytes	6 dígitos decimales de precisión.
float8	8 bytes	15 dígitos decimales de precisión.
numeric	variable	sin límite
decimal	variable	sin límite
serial	4 bytes	1 a 2147483647
bigserial	8 bytes	1 a 9223372036854775807

Tipos fecha

Tipo fecha	Tamaño	Descripción
timestamp[(p)] con zona horaria	8 bytes	Ambos fecha y hora.
timestamp[(p)][sin zona horaria]	8 bytes	Ambos fecha y hora.
interval[(p)]	12 bytes	Para intervalos de tiempo.
date	4 bytes	Solo fecha.
time[(p)][sin zona horaria]	8 bytes	Hora del día.
time[(p)] con zona horaria	12 bytes	Hora del día.

Tipos alfanuméricos

Tipo Numérico	Rango
character(n), char(n)	Longitud fija rellena con espacios en blanco.
character varying(n), varchar(n)	Longitud Variable con límite.
text	Longitud Variable ilimitada.

Otros tipos

Tipo	Descripción
bool	Tipo lógico (trae/false)
box	Caja rectangular en plano 2D
cidr	IPv4

circle	Circulo en plano 2D
line	Línea infinita en plano 2D
lseg	Segmento de línea en plano 2D
money	US-style currency
point	Punto geométrico en plano 2D
serial	Id único para índices y referencias cruzadas

8.3 Administración de una base de datos en PostgreSQL

Si el programa postmaster de Postgres está cargado y en ejecución, se pueden crear bases de datos y administrarlas.

8.3.1 Acceso a la base de datos

Una vez construida la base de datos, podemos acceder a ella utilizando el comando *psql.*

Sintaxis: ***psql*** nombredb

Salida:

```
Welcome to the Postgres interactive sql monitor

type \? for help on slash commands
type \q to quit
type \g or terminate with semicolon to execute query
You are currently connected to the database: nombredb

nombredb=>
```

Opciones:

nombredb=> `\h`	Muestra la ayuda sobre la sintaxis de varias órdenes SQL de Postgres.
nombredb=> `\i archivo`	Se utiliza para leer las consultas de un archivo, en lugar de introducirlas interactivamente.
nombredb=> `\i archivo`	Salir de psql y volver a Linux.

8.3.2 Destrucción de una base de datos

Si son administradores de la base de datos *nombrebd,* se puede destruirla utilizando el comando *dropdb.* Esta acción elimina físicamente todos los archivos asociados con la base de datos y no puede ser invertida.

Sintaxis: ***dropdb*** nombredb

8.3.3 Copia de seguridad y restauración

Las copias de seguridad deben hacerse regularmente. PostgreSQL proporciona dos utilidades para realizar las copias de seguridad: *pg_dump* para copias de seguridad de bases de datos individuales y *pg_dumpall* para realizar copias de seguridad de toda la instalación de una sola vez.

La copia de seguridad de una sola base de datos puede realizarse usando la siguiente instrucción:

```
pg_dump nombredb > nombredb.pgdump
```

Y puede ser restaurada con la siguiente instrucción

```
cat nombredb.pgdump | psql nombredb
```

8.3.4 Bases de datos grandes

Postgres permite tablas de mayor tamaño que el permitido por el sistema de archivos, puede resultar problemático el volcado de una tabla a un archivo, ya que el archivo resultante seguramente superará el tamaño máximo permitido.

Como *pg_dump* escribe en la salida estándar se pueden utilizar algunas utilerías para poder solucionar estos problemas.

Para comprimir un respaldo de una base de datos se puede hacer de la siguiente manera:

```
pg_dump nombredb | gzip > nombrearchivo.dump.gz
```

Y puede ser restaurada con:

```
createdb nombredb
gunzip –c nombrearchivo.dump.gz | psql nombredb
```

O también con:

```
cat nombrearchivo.dump.gz | gunzip | psql nombredb
```

8.4 Construcción de bases de datos con PostgreSQL

Para crear una base de datos se puede hacer con el comando *createdb,* PostgreSQL permite crear cualquier número de bases de datos en una máquina dada. Los nombres de las bases de datos han de comenzar por un caracter alfabético, y su longitud está limitada a 31 caracteres.

Sintaxis: ***createdb*** nombredb

8.4.1 Creación de tablas

La sintaxis básica para la construcción de una tabla es la siguiente:

Sintaxis:
```
create table nombre_tabla
(
      nombre_campo_1 tipo_1
      nombre_campo_2 tipo_2
      nombre_campo_n tipo_n
      key(campo_x,...)
)
```

Ejemplo:

```
create table productos
(
      prod_id integer PRIMARY KEY,
      prod_nombre text,
      prod_precio numeric DEFAULT 9.99
)
```

8.4.2 Constraints

La cláusula constraint permite definir una restricción en la base de datos, a cada restricción se le asigna un nombre que se utiliza para identificarla y para poder eliminarla cuando se quiera. Existen diferentes tipos de constraints, a continuación se describen los más importantes.

Check constraint

Ejemplo:

```
create table productos
(
      prod_id integer,
      prod_nombre text,
      prod_precio numeric CHECK (prod_precio > 0)
)
```

Descripción:

Este constraint sólo permite que se inserten valores mayores a 0 en el campo *prod_precio.*

Constraint no nulo

Ejemplo:

```
create table productos
(
      prod_id integer NOT NULL,
      prod_nombre text NOT NULL,
      prod_precio numeric
)
```

Descripción:

Este constraint no permite valores nulos en los campos *prod_id* y *prod_nombre.*

Constraint únicos

Ejemplo:

```
create table productos
(
      prod_id integer UNIQUE,
      prod_nombre text,
      prod_precio numeric
)
```

Descripción:

Unique evita que se inserten valores repetidos en el campo *prod_id.*

Constraints de llaves primarias

Ejemplo:

```
create table productos
(
      prod_id integer PRIMARY KEY,
      prod_nombre text,
      prod_precio numeric
)
```

Descripción:

Este constraint permite definir los campos que formaran la llave primaria, en este ejemplo la llave primaria es el campo *prod_id.*

Constraints de llaves foráneas

Ejemplo:

```
create table ordenes
(
      orden_id integer PRIMARY KEY,
      prod_id integer REFERENCES productos(prod_id),
      cantidad numeric
)
```

Descripción:

En el ejemplo se definen los campos que serán llaves foráneas, en el ejemplo *prod_id* es la llave foránea.

Las instrucciones de SQL para insertar, actualizar, eliminar registros, se aplican perfectamente en PostreSQL. Veamos el caso de la inserción y actualización de datos.

8.4.3 Insertando datos

Esta es la sintaxis básica para insertar un nuevo registro en una tabla.

Sintaxis:

```
insert into nombre_tabla (nombre_campo1, nombre_campo2,...)
values (valor_campo1, valor_campo2,...)
```

Ejemplo:

```
insert into productos(prod_id, prod_nombre,prod_precio)
values (1, queso, 9.99)
```

8.4.4 Actualizando datos

Esta es la sintaxis básica para actualizar un registro.

Sintaxis:

```
update      nombre_tabla
set         nombre_campo1=valor_campo1,
            nombre_campo2=valor_campo2,...
where       condiciones_de_selección
```

Ejemplo:

```
update productos set precio=10 where precio=5
```

Descripción:

En este ejemplo los valores que sean iguales a 5 los actualizará a 10.

8.5 Acceso a bases de datos PosgreSQL con PHP

El sistema de acceso y manipulación de bases de datos desde PHP es similar al de otros lenguajes de script: establece la conexión con la base de datos, ejecuta las sentencias de consulta o modificación y finalmente cierra la conexión.

Conexión con bases de datos PostgreSQL

Para establecer la conexión con una base de datos PostgreSQL desde PHP, se utiliza la función:

```
pg_connect("nombrehost basedatos usuario contraseña")
```

Los parámetros que recibe como una cadena única, indican el nombre del servidor -o IP del mismo- 'nombrehost' donde se encuentra la base de datos, el nombre de la base de datos 'basedatos', el 'usuario' de acceso a la base de datos, y la 'contraseña' de acceso. En caso de éxito la función devuelve un identificador del enlace con el sistema de bases de datos.

Finalmente, para cerrar la conexión utilizamos:

```
pg_close($conexion)
```

A esta función se le pasa como parámetro el enlace con la conexión inicialmente establecida.

Consultas sobre bases de datos PostgreSQL

Para efectuar consultas sobre una base de datos PostgreSQL, se utiliza en PHP la función:

```
pg_query($conexión $sql)
```

Esta función toma como parámetros, el enlace con la base de datos y una cadena con la consulta SQL a ejecutar (SELECT, INSERT, DELETE, etc.). Devuelve un identificador del resultado en caso de éxito o FALSE en caso de error en la consulta.

Con la ejecución de la consulta sobre la base de datos, no se puede presentar el resultado de la misma. Para poder mostrar información resultante de una

consulta se debe hacer uso de funciones complementarias. Una de las posibles es:

```
pg_fetch_array($id_resultado $fila)
```

Esta función devuelve la fila '$fila' en forma de arreglo con el resultado de la sentencia extraída identificada por el parámetro '$id_resultado'.

Ejemplo:

```
<?
      $conn = pg_pconnect("","","","","usuarios");

      if(!$conn)
      {
         echo "Ocurrió un error.\n";
         exit;
      }

      $resultado = pg_exec ($conn, "SELECT * FROM usuarios");

      if(!$resultado)
      {
         echo "Ocurrió un error.\n";
         exit;
      }

      $arr = pg_fetch_array ($resultado, 0);
      echo $arr[0] . " <- array\n";

      $arr = pg_fetch_array ($resultado, 1);
      echo $arr["nombre"] . " <- array\n";
?>
```

De esta manera se interactúa con nuestra base de datos para obtener los resultados deseados.

PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl, PHP y Python).

9. Caso Práctico (Sistema de Administración de Clientes y Servicios)

9.1 Motivación del proyecto (Situación actual)

La empresa IDEV se dedica al desarrollo de sitios Web y a ofrecer servicios de alojamiento como principales actividades.

Otra de las actividades no menos importante de la empresa, es la venta de equipo de cómputo en línea.

La visión actual de IDEV es ofrecer a sus clientes servicios en línea como pueden ser: cotizaciones, venta de productos, consulta de pagos de servicios (alojamiento) de una manera ágil y eficiente.

Con esto se busca mejorar la atención de sus clientes, así como facilitar el proceso de venta y ampliar el mercado.

Actualmente no se cuenta con un sistema con estas características.

9.2 Declaración de la visión

¿Qué se va a hacer?

Diseñar y desarrollar un sistema denominado “Sistema de Administración de Clientes y Servicios (SACS)” que permita a los administradores controlar los pedidos, almacén, clientes, y ventas.

¿Mediante qué?

El desarrollo de un sistema que automatice y mejore los procesos de recepción de pedidos y el control de los mismos. Así como una actualización constante de los catálogos y operaciones realizadas por el administrador.

¿Para qué?

Para ofrecer a los usuarios (administradores), nuevos mecanismos de atención de pedidos más eficientes, más sencillos y con una apariencia más presentable, además de obtención de información más útil.

9.3 Alcance

Se contempla que el nuevo SACS realice lo siguiente:

Registro de usuarios. El usuario podrá registrarse en línea obteniendo un número de cliente y una contraseña con la que podrá realizar cotizaciones y pedidos en línea.

Envío de pedidos. El usuario podrá realizar la cotización de una PC o de productos y realizar el pedido de los mismos.

Catálogos del sistema. El sistema contará con un módulo en el cuál se podrán tener actualizados los catálogos con los cuales funciona el sistema, tales como: clientes, productos, servicios.

Consulta de servicios. El usuario podrá consultar en línea los servicios que tiene contratados así como sus pagos y adeudos.

Exclusiones

No se contará con un módulo de pagos en línea.
No se pedirá al usuario datos relativos a cuentas de banco ni tarjetas

9.4 Análisis del Sistema de Administración de Clientes y Servicios (SACS)

El Sistema de Administración de Clientes y Servicios es un sistema en Web que permite el registro de cotizaciones y clientes, así como la administración de los catálogos.

En el sistema se encuentran identificados los siguientes actores:

- Administrador
- Cliente
- Personal

Y los siguientes casos de uso:

Para el **cliente**:

- Realizar Registro
- Registrar Autenticación
- Cotizar Productos
- Consultar pagos.
- Realizar pedido.
- Consultar productos

Para el **administrador**:

- Registrar Autenticación
- Administrar Catálogos
- Generar Reportes
- Administrar contrato de alojamiento

Para el **personal**:

- Registrar Autenticación
- Consultar pedidos
- Cotizar productos
- Realizar pedido
- Administrar catálogo de clientes
- Administrar contrato de alojamiento

9.5 Relación de requerimientos del área usuaria

Registro de clientes:
A través del sistema se podrán registrar los clientes y generar un número de cliente y una contraseña.

Realizar cotizaciones:
El cliente podrá realizar cotizaciones en línea de PC's y productos.

Envío de pedidos:
El sistema contará con un módulo en el cual se pueden mostrar los pedidos que han sido realizados por los clientes y ser atendidos.

Administración de catálogos:
El administrador del sistema por medio de este módulo podrá mantener actualizados los catálogos de clientes, productos y servicios.

Generar reportes de ventas y artículos en existencia:
El administrador podrá generar reportes sobre las ventas realizadas y los artículos que se encuentran en el almacén.

Consulta de pagos y adeudos:
Se contará con un módulo mediante el cual el usuario podrá consultar los pagos realizados sobre los servicios contratados.

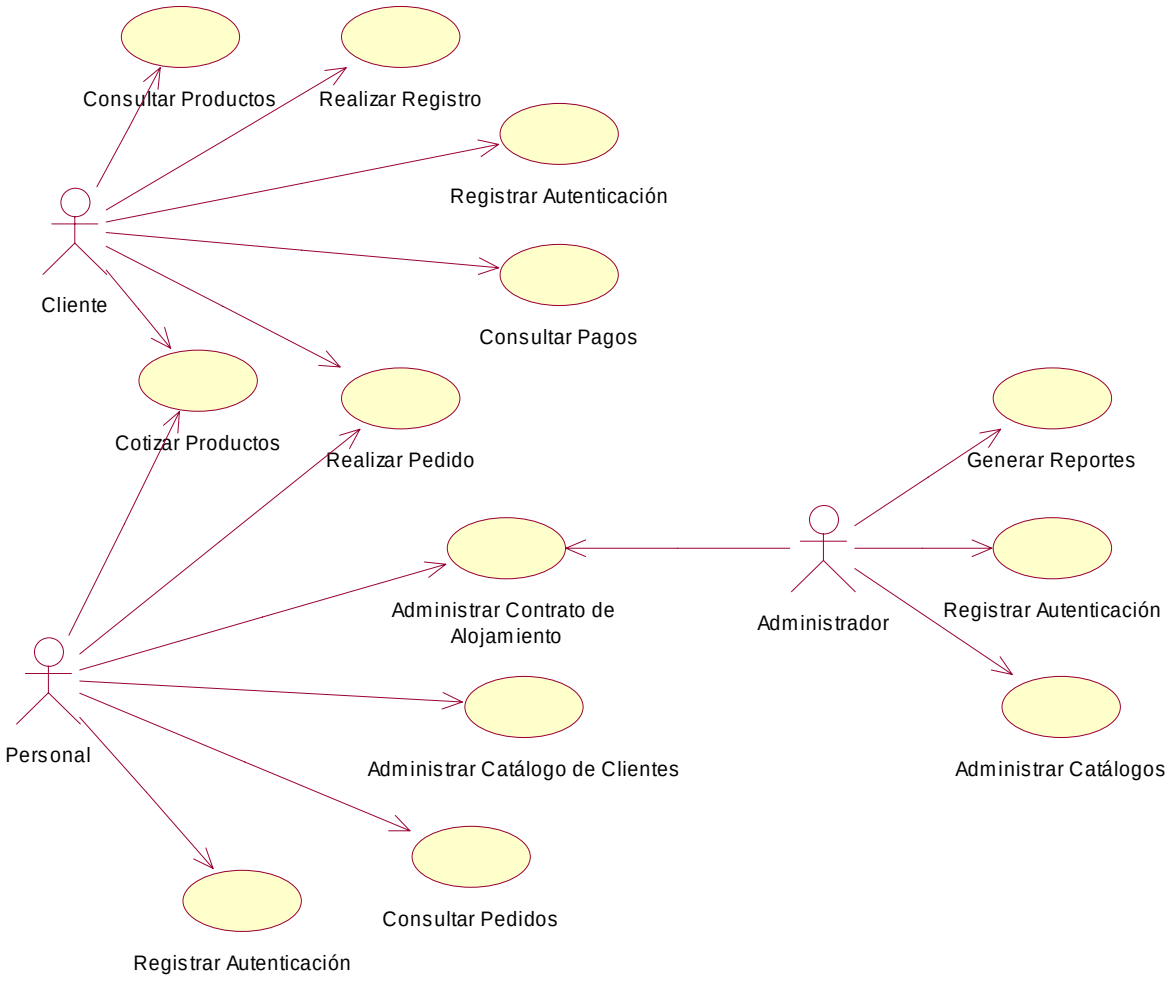
9.6 Descripción de los actores

Cliente: Es la persona que realiza cotizaciones, envía pedidos o verifica sus pagos o adeudos en caso de tener servicios contratados.

Personal: Es la persona que atenderá los pedidos realizados por los clientes, tendrá acceso al catálogo de clientes y puede realizar cotizaciones, pedidos y registrar los servicios de alojamiento contratados por los clientes.

Administrador: Es el usuario que se encarga de toda la gestión del sistema. Entre las actividades que realiza son: administrar catálogos, generar reportes.

9.7 Diagrama de Casos de uso



9.7.1 Documento de casos de uso

Introducción

Objetivo del Documento

El presente documento describe los Casos de Uso identificados para el Sistema de Administración de Clientes y Servicios (en adelante referido como SACS); sistema informático con el que se automatizará el registro y seguimiento de los pedidos de los clientes enviados desde la página Web de IDEV.

Casos de Uso

Un problema frecuente durante las etapas de análisis y diseño de un sistema es la deficiente comunicación entre los usuarios finales y el equipo de desarrollo. Es por esto conveniente implementar mecanismos que sirvan de interfase entre el lenguaje propio del dominio del negocio y un lenguaje que el equipo de desarrollo pueda entender, con el objetivo de lograr una especificación que cubra los requerimientos de los usuarios finales. Una de estas herramientas son los Casos de Uso, que forman parte del Lenguaje de Modelado Unificado (UML, por sus siglas en inglés).

Se pueden ver a los Casos de Uso como la colección de situaciones respecto al uso del sistema que se va a desarrollar. Con ellos es posible describir una secuencia de eventos, misma que es iniciada por una persona, otro sistema, una parte del hardware o por el paso del tiempo. A las entidades que inician secuencias se les conoce como actores. El resultado de la secuencia debe ser algo utilizable ya sea por el actor que la inició o cualquier otro. En resumen, los Casos de Uso son una descripción del sistema desde el punto de vista del usuario final.

Es importante señalar que los Casos de Uso no especifican la forma en que se han de resolver los problemas, sino cuáles problemas serán resueltos por el sistema es decir, cubre el qué y no el cómo.

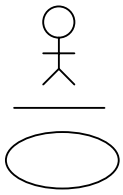
Diagramas de Casos de Uso

Una de las características más poderosas de los diagramas de Casos de Uso es su simplicidad, ya que están compuestos básicamente de tres símbolos:

Actores:

Relaciones:

Casos de Uso:



9.8 Casos de uso relativos al cliente

9.8.1 Realizar registro

CASO DE USO	**1. Realizar Registro**
BREVE DESCRIPCIÓN	El caso de uso **Realizar Registro** permite al **cliente** registrarse en IDEV, para cotizar productos, enviar pedidos, consultar sus pagos y adeudos de los servicios contratados.
ACTOR QUE INICIA	**Cliente**
PRECONDICIONES	*Del **proceso:*** El cliente debe entrar al sitio Web de IDEV y entrar al módulo de registro por medio del URL. *Del **sistema:*** Ninguno.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	
1. El **cliente** entra al módulo de registro. **2.** El **sistema** presenta un formulario con los siguientes datos: -Nombre del cliente* -Apellidos (apellido paterno, apellido materno) -RFC -Dirección* (calle*, número, colonia*, cp*, delegación*) -Correo electrónico* -Teléfono* **3.** El **cliente** captura sus datos y los envía. **4.** El **sistema** valida los datos enviados por el cliente **(E1)**. El sistema genera un número de cliente y una contraseña, envía por correo electrónico el número de cliente y la contraseña. Finalmente el sistema muestra un mensaje: **"Se le ha enviado un correo con el número de cliente y la contraseña"**.	
FLUJOS DE EXCEPCIÓN	**(E1) Los datos no son válidos:** Si el usuario deja algún campo obligatorio (*) vacío, el sistema mandará un mensaje de error que indique ***"El campo es obligatorio"***. Si el usuario ingresa algún tipo de dato no válido, el sistema mandará un mensaje de error que indique ***"El valor del campo es incorrecto"***.
POSCONDICIONES	Se genera un número de cliente y una contraseña para identificar en adelante al cliente y poder realizar diversas actividades.

9.8.2 Registrar Autenticación

CASO DE USO	**2. Registrar Autenticación**
BREVE DESCRIPCIÓN	El caso de uso **Registrar autenticación** permite al cliente autenticarse en el sistema para poder enviar un pedido o consultar los pagos realizados respecto de los servicios contratados.
ACTOR QUE INICIA	**Cliente**
PRECONDICIONES	*Del* ***proceso:*** El cliente debe contar con un número de cliente y una contraseña. *Del* ***sistema:*** La base de datos debe tener información de los clientes.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	
1. El **cliente** entra al módulo de registro. **2.** El **sistema** solicita su número de cliente* y su contraseña*. **3.** El **cliente** introduce su número de cliente y su contraseña. **4.** El **sistema** valida los datos enviados por el cliente **(E1)**, muestra un mensaje de bienvenida y un menú con las siguientes opciones: **-Realizar pedidos.** **-Consultar pagos.** **NOTA:** Cada opción será detallada en los casos de uso **"Realizar Pedidos"** y **"Consultar Pagos"** respectivamente.	
FLUJOS DE EXCEPCIÓN	**(E1) los datos del cliente son incorrectos:** Si el número de cliente no es válido el sistema mostrará el siguiente mensaje: **"El número de cliente es incorrecto".** Si la contraseña no es válida el sistema mostrará el siguiente mensaje: **"Contraseña incorrecta".** Si el cliente no capturó alguno de los dos campos el sistema mostrará el siguiente mensaje: **"El campo es obligatorio".**
POSCONDICIONES	El cliente genera una sesión en el sistema.

9.8.3 Cotizar productos

CASO DE USO	**3. Cotizar Productos**
BREVE DESCRIPCIÓN	El cliente entra a éste módulo y genera una cotización de PC o productos.
ACTOR QUE INICIA	**Cliente**
PRECONDICIONES	*Del* ***proceso:*** El cliente debe contar con un número de cliente y una contraseña, además debe iniciar una sesión en el sistema. *Del* ***sistema:*** La base de datos debe tener la información del catálogo de productos y clientes.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	

1. El **cliente** entra al módulo de cotizaciones.

2. El **sistema** muestra un menú con las siguientes opciones:

-**Cotizar PC.**
-Cotizar productos.

3. El **cliente** elige una opción:

3.1 Para la opción **"Cotizar PC":**

El **sistema** genera una lista de todos los modelos de ***motherboard**** disponibles en el almacén **(E1)**.

El **cliente** selecciona una motherboard.

El **sistema** registra el componente, genera y muestra la lista de ***procesadores**** compatibles **(E2)**.

El **cliente** selecciona un procesador.

El **sistema** registra el componente, genera y muestra la lista de ***memorias RAM**** compatibles **(E3)**.

El **cliente** selecciona una memoria de la lista.

El **sistema** registra el componente, genera y muestra la lista de ***discos duros**** compatibles **(E4)**.

El **cliente** selecciona un disco duro de la lista.

El **sistema** registra el componente, genera y muestra lista de ***tarjetas de video**** compatibles **(E5)**.

El **cliente** selecciona una tarjeta de video de la lista o selecciona la que viene integrada en la motherboard.

El **sistema** registra el componente, genera y muestra la lista de ***tarjetas de sonido*** compatibles **(E6)**.

El **cliente** selecciona una tarjeta de sonido de la lista o la que viene integrada en la motherboard.

El **sistema** registra el componente, genera y muestra la lista de ***tarjetas de red*** compatibles **(E7)**.

El **cliente** selecciona una tarjeta de red de la lista o la que viene integrada en la motherboard.

El **sistema** registra el componente, genera y muestra la lista de ***floppy*** **(E8)**.

El **cliente** selecciona un modelo de floppy o elige la opción ***"sin floppy"***.

El **sistema** registra el componente, genera y muestra la lista de la primera ***unidad de lectura*** **(E9)**.

NOTA: Las unidades de lectura pueden ser CD-ROM, CD/RW, DVD ROM, DVD/RW.

El **cliente** selecciona la primera unidad de lectura o elige la opción **"sin unidad de lectura"**.

El **sistema** registra el componente, genera y muestra la lista de la segunda ***unidad de lectura*** **(E9)**.

El **cliente** selecciona la segunda unidad de lectura o elige la opción **"sin unidad de lectura"**.

El **sistema** registra el componente, genera y muestra la lista de ***gabinetes*** **(E10)**.

El **cliente** selecciona un gabinete.

El **sistema** registra el componente y genera la lista de ***monitores*** **(E11)**.

El **cliente** selecciona un monitor o elige la opción ***"sin monitor"***.

El **sistema** registra el componente, muestra y genera lista de ***teclados*** **(E12)**.

El **cliente** selecciona un teclado o elige la opción **"sin teclado"**.

El **sistema** registra el componente, muestra y genera lista de los ***"mouse"*** **(E13)**.

El **cliente** selecciona un mouse o elige la opción **"sin mouse"**.

El **sistema** registra el componente, calcula el costo total y muestra los detalles de la cotización.

3.2 Para la opción **Cotizar productos:**

El **sistema** muestra el catálogo de productos ordenados en las siguientes categorías:

-Procesadores
-Motherboards
-Gabinetes
-Memorias RAM
-Unidades ROM
-Unidades Floppy
-Memorias FLASH
-Discos Duros
-Monitores
-Tarjetas de red
-Tarjetas de video
-Tarjetas de sonido
-Impresoras
-Teclados
-Ratones
-Cables
-Fax módems
-Productos de limpieza
-Software

El **cliente** selecciona alguna de las categorías.

El **sistema** muestra todos los productos de esa categoría (ordenados a su vez en subcategorías) disponibles en el almacén, por cada registro mostrará un campo para introducir la cantidad y una opción para agregarlo a la cotización **(E14)**.

El **cliente** selecciona un producto e introduce la cantidad que desea adquirir, finalmente agrega el producto a su cotización.

El **sistema** valida la cantidad introducida por el cliente **(E15).**

El **sistema** verifica si existe la cantidad de ese producto en el almacén **(E16).** El **sistema** registra el producto, muestra la cotización actual, y los detalles de todos los productos agregados (cantidad, producto, precio de lista e importe y el total). **NOTA:** El cliente puede agregar **n** cantidad de productos a la cotización. Cada que se agregue un nuevo producto, aparecerán los detalles y la opción de eliminar cada producto agregado, y el total.	
FLUJOS DE EXCEPCIÓN	**(E1) No hay motherboard en el almacén.** Si este producto está agotado, el sistema no puede generar la lista y por lo tanto no se podrá realizar una cotización. El sistema mostrará el siguiente mensaje **"La cotización no puede realizarse porque las motherboards están agotadas".** **(E2) No hay procesadores compatibles en el almacén.** Si este producto está agotado, el sistema no puede generar la lista y por lo tanto no se podrá realizar una cotización. El sistema mostrará el siguiente mensaje **"La cotización no puede realizarse porque no hay procesadores disponibles en este momento".** **(E3) No hay memorias RAM compatibles en el almacén.** Si este producto está agotado, el sistema no puede generar la lista y por lo tanto no se podrá realizar una cotización. El sistema mostrará el siguiente mensaje **"La cotización no puede realizarse porque no hay memorias RAM disponibles en este momento".** **(E4) No hay discos duros compatibles en el almacén.** Si este producto está agotado, el sistema no puede generar la lista y por lo tanto no se podrá realizar una cotización. El sistema mostrará el siguiente mensaje **"La cotización no puede realizarse porque no hay discos duros disponibles en este momento".**

	(E5) No hay tarjetas de video compatibles en el almacén. Si este producto está agotado y la motherboard no tiene tarjeta de video integrada, el sistema no puede generar la lista y por lo tanto no se podrá realizar una cotización. El sistema mostrará el siguiente mensaje **"La cotización no puede realizarse porque no hay tarjetas de video disponibles en este momento y la motherboard no tiene una integrada".** **(E6) No hay tarjetas de sonido compatibles en el almacén.** Si este producto está agotado y la motherboard no tiene una tarjeta de sonido integrada, el sistema mostrará el mensaje **"No hay tarjetas de sonido disponibles en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E7) No hay tarjetas de red compatibles en el almacén.** Si este producto está agotado y la motherboard no tiene una tarjeta de red integrada, el sistema mostrará el mensaje **"No hay tarjetas de red disponibles en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E8) No hay floppy en el almacén.** Si este producto está agotado, el sistema mostrará el mensaje **"No hay floppy disponible en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E9) No hay unidad de lectura en el almacén.** Si este producto está agotado, el sistema mostrará el mensaje **"No hay unidad de lectura disponible en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E10) No hay gabinetes en el almacén.** Si este producto está agotado el sistema mostrará el mensaje **"La cotización no puede realizarse porque no hay gabinetes disponibles en este momento"**.

	(E11) No hay monitores en el almacén. Si este producto está agotado, el sistema mostrará el mensaje **"No hay monitores disponibles en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E12) No hay teclados en el almacén.** Si este producto está agotado, el sistema mostrará el mensaje **"No hay teclados disponibles en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E13) No hay mouse en el almacén.** Si este producto está agotado, el sistema mostrará el mensaje **"No hay mouse disponibles en este momento".** La cotización puede continuar ya que no es indispensable este componente. **(E14) El producto está agotado.** Si el producto está agotado el sistema quita el campo para introducir la cantidad y la opción para agregarlo a la cotización. **(E15) La cantidad no es válida.** Si la cantidad no es válida, el **sistema** muestra un mensaje de error **"Cantidad no válida"**. **(E16) No existe la cantidad del producto en el almacén.** El sistema mostrará un error **"Cantidad no disponible"** y muestra la lista de productos y la cotización actual.
POSCONDICIONES	**Se genera un número de cotización**

9.8.4 Consultar Pagos

CASO DE USO	**4. Consultar Pagos**
BREVE DESCRIPCIÓN	El cliente entra a este módulo y consulta los pagos o adeudos de los servicios contratados.
ACTOR QUE INICIA	Cliente
PRECONDICIONES	*Del **proceso:*** El cliente debe contar con un número de cliente y una contraseña, además debe iniciar una sesión en el sistema. *Del **sistema:*** La base de datos debe tener información de clientes y servicios de alojamiento.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	
1. El **cliente** entra al módulo de consulta de pagos. **2.** El **sistema** consulta si el usuario tiene servicios contratados **(E1)**. El sistema genera el siguiente informe: -Numero de cliente. -Nombre -Detalles del Paquete de alojamiento que contrató. -Fecha de activación. -Fecha de vencimiento. -Tiempo de alojamiento. -Cantidad total a pagar. Numero de pagos realizados: -Numero -Fecha -Cantidad Adeudo: -Cantidad a pagar -Fecha límite para pagar. **Nota:** Se mostraran todos los pagos realizados por el cliente.	
FLUJOS DE EXCEPCIÓN	**(E1) El cliente no tiene servicios contratados.** Si el cliente no tiene servicios contratados el sistema mostrará un mensaje **“No tiene servicios contratados”**.
POSCONDICIONES	No existen poscondiciones

9.8.5 Realizar pedido

CASO DE USO	**5. Realizar pedido**
BREVE DESCRIPCIÓN	El cliente entra a este módulo para enviar al personal la cotización que realizó para que ésta sea atendida.
ACTOR QUE INICIA	**Cliente**
PRECONDICIONES	*Del **proceso**:* El cliente debe tener un número de cliente y una contraseña, debe iniciar una sesión en el sistema y realizar una cotización de PC o productos. *Del **sistema**:* La base de datos debe tener información de los clientes y productos.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	
1. El **cliente** selecciona la opción **"Enviar pedido"**. **2.** El **sistema** mostrará un cuadro de dialogo para la confirmación del pedido **(E1)**. **3.** El **cliente** confirma el pedido. **4.** El **sistema**, genera un número de pedido, registra los detalles de la cotización y envía por correo electrónico los datos del pedido al personal para que sea atendido. Finalmente envía un mensaje al cliente con los siguientes datos: -Numero de pedido -Numero de cliente -Fecha Detalle de cada producto adquirido: -Clave -Cantidad -Descripción -Precio -TOTAL	
FLUJOS DE EXCEPCIÓN	**(E1) El cliente cancela el envió de pedido** Si el cliente selecciona cancelar pedido, el **sistema** mostrará la cotización actual.
POSCONDICIONES	El sistema genera número de pedido.

9.8.*6* Consultar productos

CASO DE USO	**6. Consultar productos**
BREVE DESCRIPCIÓN	El cliente entra a este módulo para consultar los productos a la venta en IDEV.
ACTOR QUE INICIA	**Cliente**
PRECONDICIONES	*Del **proceso:*** Entrar al módulo de consulta de productos en el sitio Web de IDEV. *Del **sistema:*** La base de datos debe tener información de los productos.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	
1. El **cliente** entra al módulo de consulta de productos. **2.** El **sistema** muestra el catálogo de productos ordenados en las siguientes categorías: -Procesadores -Motherboards -Gabinetes -Memorias RAM -Unidades ROM -Unidades Floppy -Memorias FLASH -Discos Duros -Monitores -Tarjetas de red -Tarjetas de video -Tarjetas de sonido -Impresoras –Teclados –Ratones -Fax módems -Software **3.** El **cliente** selecciona alguna de las categorías. El **sistema** muestra todos los productos de esa categoría (ordenados a su vez en subcategorías) disponibles en el almacén **(E1)**, por cada registro mostrará los detalles del producto (clave, descripción, precio, etc.).	
FLUJOS DE EXCEPCIÓN	**(E1) El artículo está agotado** Si algún producto está agotado, el sistema agregará la leyenda **"Artículo agotado"** al registro.
POSCONDICIONES	**No existen**

9.9 Casos de uso relativos al administrador

9.9.1 Registrar autenticación

CASO DE USO	**1. Registrar Autenticación**
BREVE DESCRIPCIÓN	En este caso de uso el administrador se autentica en el sistema para poder administrar los catálogos y generar reportes.
ACTOR QUE INICIA	Administrador
PRECONDICIONES	*Del* ***proceso:*** Contar con un nombre de usuario y una contraseña. *Del* ***sistema:*** La base de datos debe tener información del personal.
ACTOR QUE SE BENEFICIA	Administrador
DESCRIPCIÓN PASO A PASO	
1. El **administrador** entra al módulo de registro de autenticación. **2.** El **sistema** solicita su nombre de usuario y su contraseña. **3.** El **administrador** introduce su nombre de usuario y contraseña. **4.** El **sistema** valida los datos enviados por el administrador **(E1)**. El **sistema** envía un mensaje de bienvenida y muestra un menú con las siguientes opciones: **-Administrar catálogos.** **-Generar reportes.**	
FLUJOS DE EXCEPCIÓN	**(E1) Los datos son incorrectos.** Si el nombre de usuario no es válido, el sistema mostrará el siguiente mensaje: **"El nombre de usuario es incorrecto".** Si la contraseña no es válida, el sistema mostrará el siguiente mensaje: **"Contraseña incorrecta".** Si el administrador no capturó alguno de los dos campos el sistema mostrará el siguiente mensaje: **"El campo es obligatorio".**
POSCONDICIONES	Se genera una sesión en el sistema.

9.9.2 Administrar catálogos

CASO DE USO	**2. Administrar Catálogos**
BREVE DESCRIPCIÓN	El administrador del sistema elige un catálogo con el fin de realizar algunas operaciones de modificación tales como: altas, bajas y cambios.
ACTOR QUE INICIA	Administrador
PRECONDICIONES	*Del **proceso:*** Contar con un nombre de usuario y una contraseña, iniciar una sesión. *Del **sistema:*** La base de datos debe tener información de los usuarios del sistema, clientes, productos y servicios de alojamiento.
ACTOR QUE SE BENEFICIA	Administrador
DESCRIPCIÓN PASO A PASO	

1. El **administrador** entra al módulo de administración de catálogos.

2. El **sistema** muestra un menú con los siguientes catálogos:

-Productos
-Clientes
-Servicios de alojamiento

3. El **administrador** selecciona un catálogo:

3.1 Para el catálogo de **Productos**:

El **sistema** muestra un menú con las siguientes opciones:

-Registrar nuevo producto.
-Eliminar productos.
-Actualizar productos.

El **administrador** selecciona alguna opción:

Para la opción **“Registrar nuevo producto”,** el **sistema** muestra un formulario con los siguientes datos:

-Clave del producto*
-Categoría*
-Subcategoría
-Marca*
-Detalles
-Descripción*
-Precio*
-Cantidad en existencia*

El **administrador** captura los datos del nuevo producto.

El **sistema** valida los datos capturados por el administrador **(E1)**. El **sistema** registra el producto y muestra los detalles del producto agregado (clave, categoría, etc.).

Para la opción **"Eliminar producto"** el **sistema** muestra el catálogo de productos ordenados por categorías.

El **administrador** selecciona alguna de las categorías de productos.

El **sistema** muestra los productos de esa categoría (ordenados a su vez en subcategorías) y por cada producto muestra la opción **"Eliminar producto"**.

El **administrador** selecciona el producto que quiere eliminar.

El **sistema** pide la confirmación para eliminar el producto **(E2)**.

El **administrador** confirma la eliminación del producto.

El **sistema** muestra los detalles del producto eliminado.

Para la opción **"Actualizar Productos"** el **sistema** muestra el catálogo de productos ordenados por categorías.

El **administrador** selecciona alguna de las categorías de productos.

El **sistema** muestra los productos de esa categoría (ordenados a su vez en subcategorías) y por cada producto muestra la opción **"Actualizar producto"**.

El **administrador** selecciona el producto que quiere actualizar.

El **sistema** pide la confirmación para actualizar el producto **(E3)**.

El **administrador** confirma la actualización del producto.

El **sistema** muestra un formulario con todos los datos actuales del producto seleccionado.

El **administrador** actualiza los datos.

El **sistema** valida los datos enviados por el administrador **(E1).** El **sistema** actualiza el producto y muestra los detalles del producto actualizado.

3.2 Para el catálogo de **Clientes**:

El **sistema** muestra un menú con las siguientes opciones:

-Capturar nuevo cliente
-Eliminar cliente
-Actualizar datos de cliente

El **administrador** selecciona alguna opción.

Para la opción **"Capturar nuevo cliente"** el **sistema** muestra un formulario con los siguientes datos:

-Nombre del cliente*
-Apellidos (apellido paterno, apellido materno)
-RFC
-Dirección* (calle*, número, colonia*, cp*, delegación*)
-Correo electrónico*
-Teléfono*

El **administrador** captura los datos del cliente.

El **sistema** valida los datos enviados por el administrador **(E4)**. El **sistema** registra los datos y muestra los detalles del nuevo cliente registrado.

Para la opción **"Eliminar cliente"** el **sistema** genera la lista de todos los clientes registrados y los ordena alfabéticamente por nombre. Además por cada registro muestra la opción **"Eliminar cliente".**

El **administrador** selecciona al cliente que desea eliminar.

El **sistema** pide la confirmación para eliminar al cliente **(E5)**.

El **administrador** confirma la eliminación.

El **sistema** elimina y muestra el registro eliminado.

Para la opción **"Actualizar datos del cliente"** el **sistema** genera la lista de clientes, por cada registro muestra la opción **"Actualizar datos".**

El **administrador** selecciona el cliente que desea actualizar.

El **sistema** muestra el formulario con todos los datos actuales del cliente.

El **administrador** actualiza los datos del cliente.

El **sistema** valida los datos enviados por el **administrador (E4).**El **sistema** actualiza los datos y muestra los detalles del cliente.

3.3 Para el catálogo de **servicios de alojamiento**:

El **sistema** muestra un menú con las siguientes opciones:

-Capturar nuevo servicio de alojamiento
-Eliminar servicio de alojamiento
-Actualizar datos de servicio alojamiento

El **administrador** selecciona alguna opción:

Para la opción **"Capturar nuevo servicio de alojamiento"** el **sistema** muestra un formulario con los siguientes datos:

-Numero de servicio*
-Tipo de plan de alojamiento*
-Tarifa mensual*
-Espacio Web*
-Dominio incluido*
-Cuentas de correo*
-Transferencia*
-Bases de datos

El **administrador** captura los datos del servicio de alojamiento.

El **sistema** valida los datos enviados por el administrador **(E6)**. El **sistema** registra los datos y muestra los detalles del nuevo servicio de alojamiento registrado.

Para la opción **"Eliminar servicio de alojamiento"** el **sistema** muestra la lista de servicios con la opción de eliminar cada registro.

El **administrador** selecciona el registro que desea eliminar.

El **sistema** pide la confirmación para eliminar el registro **(E7)**.

El **administrador** confirma la eliminación.

El **sistema** muestra el registro eliminado.

<table>
<tr><td colspan="2">Para la opción “Actualizar datos de servicio de alojamiento” el sistema muestra la lista de servicios con la opción de actualizar cada registro.

El administrador selecciona el registro que desea actualizar.

El sistema presenta el formulario con los todos los datos actuales del registro.

El administrador actualiza los datos.

El sistema valida los datos actualizados por el administrador (E6).El sistema registra los datos y muestra los detalles del nuevo servicio de alojamiento registrado.</td></tr>
<tr><td>FLUJOS DE EXCEPCIÓN</td><td>(E1) Los datos del producto son incorrectos:

Si el administrador no capturó datos obligatorios, el sistema mostrara el siguiente mensaje: “Los datos son obligatorios”.

Si el administrador capturo datos no válidos el sistema mostrará el mensaje: “Los datos son incorrectos”.

(E2) El administrador cancela la eliminación del producto.

Si el administrador cancela la eliminación del producto el sistema mostrará la última lista de productos seleccionados.

(E3) El administrador cancela la actualización del producto.

Si el administrador cancela la actualización del producto el sistema mostrará la última lista de productos seleccionados.

(E4) Los datos del cliente son incorrectos:

Si el administrador no capturó datos obligatorios, el sistema mostrara el siguiente mensaje: “Los datos son obligatorios”.

Si el administrador capturo datos no válidos el sistema mostrará el mensaje: “Los datos son incorrectos”.

(E5) El administrador cancela la eliminación del cliente.

Si el administrador cancela la eliminación del cliente, el sistema mostrará la lista de clientes.</td></tr>
</table>

	(E6) Los datos del servicio de alojamiento son incorrectos: Si el administrador no capturó datos obligatorios, el sistema mostrara el siguiente mensaje: **“Los datos son obligatorios”.** Si el administrador capturo datos no válidos el sistema mostrará el mensaje: **“Los datos son incorrectos”.** **(E7) El administrador cancela la eliminación del servicio de alojamiento.** Si el administrador cancela la eliminación del producto el sistema mostrará la última lista de productos seleccionados.
POSCONDICIONES	No existen.

9.9.3 Generar reportes

CASO DE USO	**3. Generar Reportes**
BREVE DESCRIPCIÓN	En este caso de uso el administrador del Sistema se autentica para generar un reporte.
ACTOR QUE INICIA	Administrador
PRECONDICIONES	*Del* ***proceso:*** Contar con un nombre de usuario y una contraseña, iniciar una sesión en el sistema. *Del* ***sistema:*** La base de datos debe tener información de clientes, productos y servicios de alojamiento.
ACTOR QUE SE BENEFICIA	Administrador

DESCRIPCIÓN PASO A PASO

1. El **administrador** entra al módulo de generación de reportes.

2. El **sistema** muestra un menú con las siguientes opciones:

-Reporte de clientes.
-Reporte de pedidos.
-Reporte de artículos en existencia.
-Reporte de ventas por fecha.

2. El **administrador** selecciona alguna de las opciones.

2.1 Para la opción **"Reporte de clientes"** el **sistema** genera la siguiente salida **(E1)**:

-Numero de cliente
-Nombre del cliente

-Detalles del Paquete de alojamiento
-Meses contratados
-Cantidad total a pagar

Pagos realizados:
-Numero
-Fecha
-Cantidad (Se muestran todos los pagos realizados)

Adeudo:
-Cantidad a pagar
-Fecha limite para pagar

Nota: Se mostrará un registro por cada cliente.

2.2 Para la opción **"Reporte de pedidos"** el **sistema** genera la siguiente salida **(E2)**:

-Numero de pedido
-Fecha
-Datos del cliente
-Detalles del pedido
-Cantidad

Nota: Se mostrará un registro por cada pedido. Los detalles del pedido podrán ser consultados (clave, marca, descripción, etc.).

2.3 Para la opción **"Reporte de artículos en existencia":**

El **sistema** muestra las opciones para este tipo de reporte:

-Mostrar Existencia de artículos por categoría
-Mostrar Artículos agotados

El **administrador** selecciona una opción.

Para la opción **"Existencia de artículos por categoría"** el **sistema** muestra todas las categorías de productos.

El **administrador** selecciona una categoría.

El **sistema** genera el siguiente reporte:

-Categoría
-Subcategoría
-Clave del producto
-Descripción
-Cantidad

NOTA: se genera un registro para todos los artículos de la categoría seleccionada.

Para la opción **"Mostrar artículos agotados"**

el sistema genera la siguiente salida **(E3)**:

Detalles del producto agotado:

-Categoría
-Subcategoría
-Clave del producto
-Descripción

NOTA: Se genera un registro por cada artículo agotado.

2.4 Para la opción **"Reporte de ventas por fecha"** el **sistema** muestra un formulario con los siguientes datos: fecha inicial, fecha final.

El **administrador** captura las fechas.

El **sistema** valida las fechas **(E4)**. El **sistema** muestra la siguiente salida **(E5)**:

<table>
<tr><td colspan="2">-Número de pedido
-Fecha
-Cliente
-Cantidad

NOTA: Se genera un registro por cada venta en ese rango de fechas.</td></tr>
<tr><td>FLUJOS DE EXCEPCIÓN</td><td>(E1) NO hay clientes registrados.

Si no hay clientes registrados el sistema mostrará el siguiente mensaje: “No hay clientes registrados”.

(E2) No hay pedidos registrados.

Si no hay pedidos registrados el sistema mostrará el siguiente mensaje: “No hay pedidos registrados”.

(E3) No hay artículos agotados.

Si no hay artículos agotados, el sistema mostrará el siguiente mensaje: “No hay artículos agotados”.

(E4) Fechas incorrectas.

Si las fechas son incorrectas o si la fecha inicial es mayor a la final (o lo contrario), el sistema muestra un mensaje de error “Fechas incorrectas”.

(E5) No hay ventas en ese rango de fechas.

Si no hay ventas en ese rango de fechas, el sistema mostrará el siguiente mensaje: “No hay ventas en ese rango de fechas”.</td></tr>
<tr><td>POSCONDICIONES</td><td>No existen</td></tr>
</table>

9.9.4 Administrar contrato de alojamiento

Ver caso de uso “administrar contrato de alojamiento” relativo al personal.

9.10 Casos de uso relativos al personal

9.10.1 Registrar Autenticación

CASO DE USO	**1. Registrar Autenticación**
BREVE DESCRIPCIÓN	En este caso de uso el personal se autentica en el sistema para poder realizar cotizaciones, consultar pedidos, realizar pedidos y administrar catálogos de clientes.
ACTOR QUE INICIA	Personal
PRECONDICIONES	*Del* ***proceso:*** Contar con un nombre de usuario y una contraseña. *Del* ***sistema:*** La base de datos debe tener información de los usuarios del sistema, clientes y servicios.
ACTOR QUE SE BENEFICIA	Personal
DESCRIPCIÓN PASO A PASO	
1. El **personal** entra al módulo de registro de autenticación. **2.** El **sistema** solicita el nombre de usuario y su contraseña. **3.** El **personal** introduce su nombre de usuario y contraseña. **4.** El **sistema** valida los datos **(E1)**. El **sistema** genera una sesión, y envía un mensaje de bienvenida. El sistema mostrará un menú con las siguientes opciones: -Consultar Pedidos -Cotizar Productos -Realizar Pedido -Administrar Catálogo de Clientes -Administrar Contrato de Alojamiento	
FLUJOS DE EXCEPCIÓN	**(E1) los datos del cliente son incorrectos:** Si el número de cliente no es válido el sistema mostrará el siguiente mensaje: **"El número de cliente es incorrecto".** Si la contraseña no es válida el sistema mostrará el siguiente mensaje: **"Contraseña incorrecta".** Si el cliente no capturó alguno de los dos campos el sistema mostrará el siguiente mensaje: **"El campo es obligatorio".**
POSCONDICIONES	El **personal** genera una sesión en el SACS.

9.10.2 Consultar Pedidos

CASO DE USO	**2. Consultar Pedidos**
BREVE DESCRIPCIÓN	El personal entra a este módulo y consulta los pedidos realizados por un cliente para que sea atendido o cancelado.
ACTOR QUE INICIA	Personal
PRECONDICIONES	*Del **proceso:*** El personal debe contar con un nombre de usuario y una contraseña, además debe iniciar una sesión en el sistema. *Del **sistema:*** La base de datos debe contar con la información de clientes, personal, productos y servicios.
DESCRIPCIÓN PASO A PASO	

1. El **personal** entra al módulo de consulta de pedidos.

2. El **sistema** muestra los pedidos que han llegado **(E1)** y por cada registro muestra los siguientes datos:

-Numero de pedido
-Fecha
-Detalles
-Número de Cliente
-Total

Además el **sistema** habilitará 2 opciones por cada pedido:

-Cancelar pedido
-Concretar pedido

3. El **personal** selecciona una opción por registro:

Para la opción **"Cancelar pedido",** el sistema envía un mensaje de confirmación **(E2)**.

El **personal** confirma la cancelación del pedido.

El **sistema** elimina el pedido y mostrará un mensaje "**Pedido cancelado**".

Para la opción **"Concretar pedido"** el sistema envía un mensaje de confirmación **(E3)".**

El **personal** confirma el pedido.

El **sistema** mostrará un formato imprimible con los siguientes datos:

-Numero de Pedido
-Datos del cliente (numero de cliente, nombre, dirección, etc.)
-Detalles del pedido (todos los productos adquiridos)
-Total

FLUJOS DE EXCEPCIÓN	**(E1) No hay pedidos por atender.** Si no hay pedidos por atender, el sistema mostrará el siguiente mensaje. **"No hay pedidos por atender".** **(E2) El personal no confirma la opción "Concretar pedido".** El sistema sigue mostrando la lista de pedidos. **(E3) El personal no confirma la opción "Cancelar pedido".** El sistema sigue mostrando la lista de pedidos.
POSCONDICIONES	No existen.

9.10.3 Cotizar Productos

CASO DE USO	**3. Cotizar Productos**
BREVE DESCRIPCIÓN	El cliente entra a éste módulo y genera una cotización de PC o productos.
ACTOR QUE INICIA	**Personal**
PRECONDICIONES	*Del* ***proceso:*** El personal debe contar con un nombre de usuario y una contraseña, además debe iniciar una sesión en el sistema. *Del* ***sistema:*** La base de datos debe tener la información del catálogo de productos, clientes y personal.
ACTOR QUE SE BENEFICIA	**Cliente**
DESCRIPCIÓN PASO A PASO	
1. El **personal** entra al módulo de cotización. **2.** El **sistema** solicita el número de cliente. **3.** El **personal** introduce el número de cliente. **4.** El **sistema** valida el número de cliente **(E1)**. El sistema muestra los datos del cliente. Continuar con el paso **2** y siguientes, del caso de uso **"Cotizar producto"** referente al cliente.	

FLUJOS DE EXCEPCIÓN	**(E1) El número de cliente es inválido.** Si el personal introduce un número de cliente incorrecto, el sistema muestra el mensaje: **"Número de cliente incorrecto".** Si el personal no captura el número de cliente, el sistema muestra el mensaje: **"El número de cliente es requerido para realizar la cotización"**.
POSCONDICIONES	El sistema genera un número de cotización.

9.10.4 Realizar Pedido

Ver caso de uso "Realizar Pedido" referente al cliente.

9.10.5 Administrar catálogo de clientes

CASO DE USO	**2. Administrar Catálogo de clientes**
BREVE DESCRIPCIÓN	El **personal** realiza algunas operaciones de modificación tales como: altas, bajas y cambios sobre el catálogo de clientes.
ACTOR QUE INICIA	Administrador
PRECONDICIONES	*Del **proceso:*** Contar con un nombre de usuario y una contraseña, iniciar una sesión. *Del **sistema:*** La base de datos debe tener información de los usuarios del sistema y clientes.
ACTOR QUE SE BENEFICIA	Personal
DESCRIPCIÓN PASO A PASO	
1. El **personal** entra al módulo de administración del catálogo de clientes. **2.** El **sistema** muestra un menú con las siguientes opciones: -Capturar nuevo cliente -Eliminar cliente -Actualizar datos de cliente 3. El **personal** selecciona alguna opción. 3.1 Para la opción **"Capturar nuevo cliente"** el **sistema** muestra un formulario con los siguientes datos:	

-Nombre* -Apellidos -RFC -Dirección* -Correo electrónico* -Teléfono* El **personal** captura los datos del cliente. El **sistema** valida los datos enviados por el administrador **(E1)**. El **sistema** registra los datos y muestra los detalles del nuevo cliente registrado. Para la opción **"Eliminar cliente"** el **sistema** genera la lista de todos los clientes registrados y los ordena alfabéticamente por nombre. Además por cada registro muestra la opción **"Eliminar cliente".** El **personal** selecciona al cliente que desea eliminar. El **sistema** pide la confirmación para eliminar al cliente **(E2)**. El **personal** confirma la eliminación. El **sistema** muestra el registro eliminado. Para la opción **"Actualizar datos del cliente"** el **sistema** genera la lista de clientes, por cada registro muestra la opción **"Actualizar datos".** El **personal** selecciona al cliente que desea actualizar. El **sistema** muestra el formulario con todos los datos actuales del cliente. El **administrador** actualiza los datos del cliente. El **sistema** valida los datos enviados por el **administrador (E1).**El **sistema** actualiza los datos y muestra los detalles del cliente.	
FLUJOS DE EXCEPCIÓN	**(E1) Los datos del cliente son incorrectos:** Si el personal no capturó datos obligatorios, el sistema mostrara el siguiente mensaje: **"Los datos son obligatorios".** Si el personal capturo datos no válidos el sistema mostrará el mensaje: **"Los datos son incorrectos".** **(E2) El personal cancela la eliminación del cliente.** Si el administrador cancela la eliminación del cliente, el sistema mostrará la lista de clientes.
POSCONDICIONES	No existen.

9.10.6 Administrar contrato de alojamiento

CASO DE USO	**6. Administrar contrato de alojamiento**
BREVE DESCRIPCIÓN	El personal registra el servicio de alojamiento contratado por un cliente.
ACTOR QUE INICIA	**Personal**
PRECONDICIONES	*Del **proceso**:* Contar con un nombre de usuario y una contraseña, iniciar una sesión en el sistema. *Del **sistema**:* La base de datos debe tener información de los clientes, usuarios del sistema, y servicios de alojamiento.
ACTOR QUE SE BENEFICIA	**Personal**

DESCRIPCIÓN PASO A PASO

1. El **personal** entra al módulo de administración de contrato de alojamiento.

2. El **sistema** muestra un menú con las siguientes opciones:

-Registrar contrato de alojamiento
-Cancelar contrato de alojamiento
-Registrar pago

3. El **personal** selecciona una opción.

3.1 Para la opción **"Registrar contrato de alojamiento"** el **sistema** presenta un formulario con los siguientes datos:

-Numero de contrato*
-Numero de cliente*
-Servicio de alojamiento*
-Meses contratados*
-Fecha de activación
-Fecha de vencimiento

El **personal** captura los datos.

El **sistema** valida los datos enviados por el personal **(E1)**. El **sistema** los registra y presenta el siguiente informe.

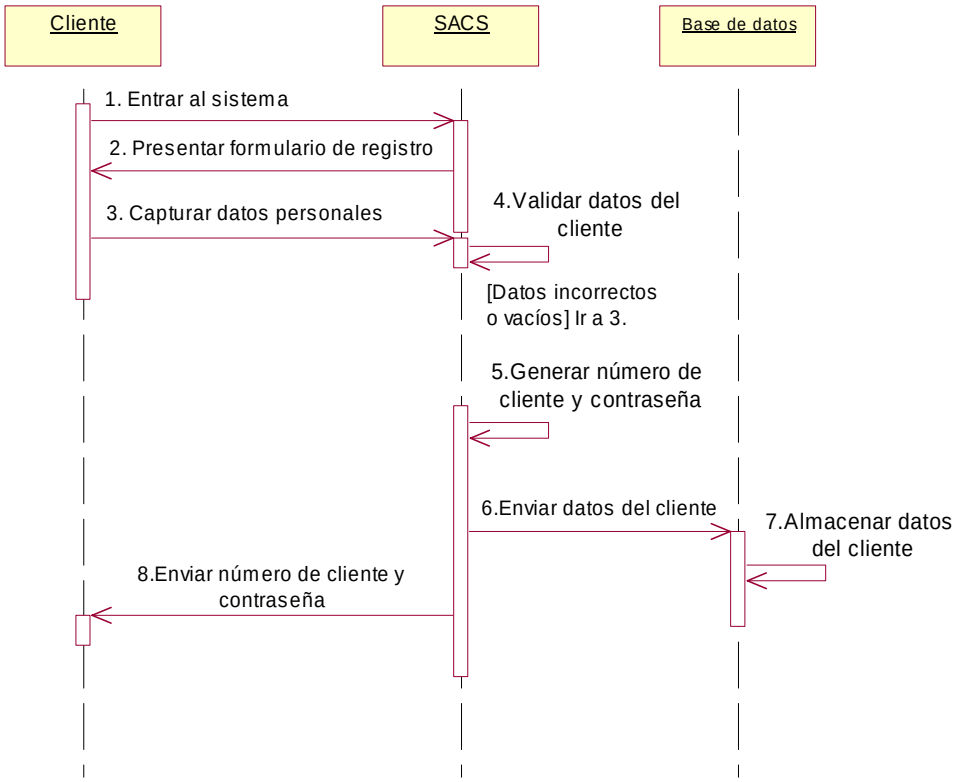
-Numero de cliente
-Nombre
-Tipo de plan de alojamiento*
-Fecha de activación
-Fecha de vencimiento
-Tarifa mensual*
-Total.

3.2 Para la opción **"Cancelar contrato de alojamiento"** el **sistema** muestra una lista con todos los contratos. Por cada contrato habilitará la opción **"Cancelar contrato".**

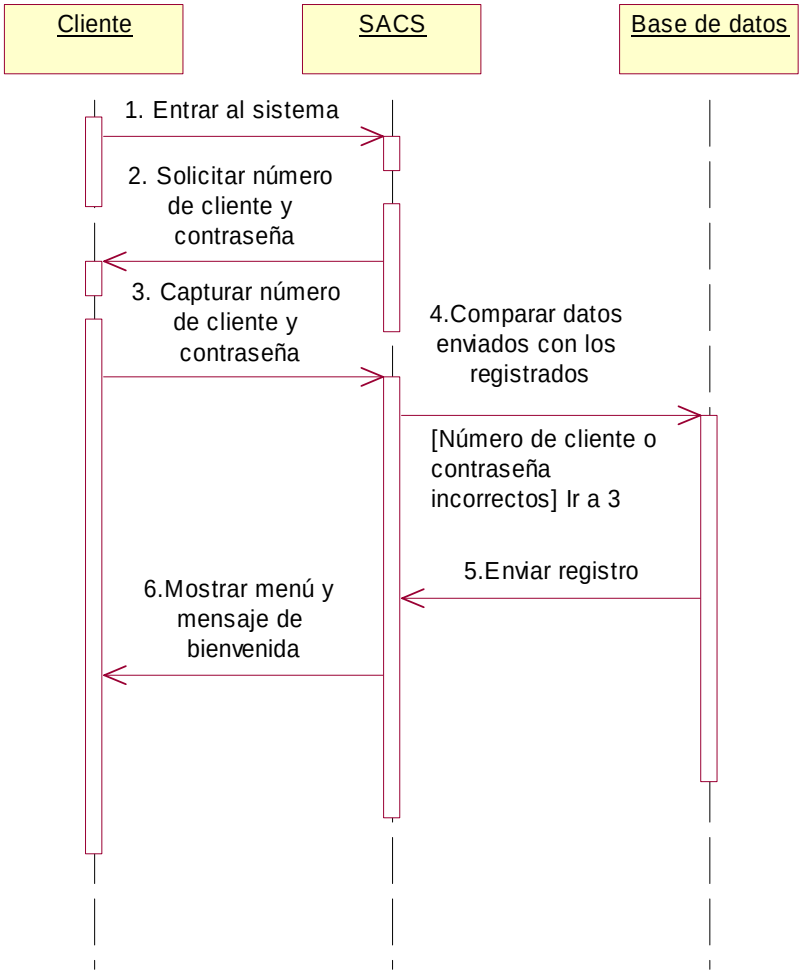
El **personal** selecciona el contrato que desea cancelar. El **sistema** solicita la confirmación para eliminar el registro **(E2)**. El **personal** confirma o cancela la eliminación del registro. El **sistema** muestra el registro eliminado o sigue mostrando la lista de contratos de alojamiento. **3.3** Para la opción **"Registrar pago"** el sistema solicita el número de cliente* y el número de contrato*. El **personal** captura el número de cliente y el numero de contrato. El **sistema** valida los datos enviados por el cliente **(E3)**. El sistema muestra el siguiente informe: -Datos del cliente (numero, nombre, etc.) -Datos del servicio contratado(tarifa mensual, espacio Web, etc.) -Datos del contrato(fecha de activación, fecha de vencimiento, etc.) -Numero de pago -Cantidad El **sistema** solicita la confirmación para registrar el pago. El **personal** confirma el registro del pago. **NOTA:** El sistema determina automáticamente el número de pago y la cantidad.	
FLUJOS DE EXCEPCIÓN	**(E1) Datos de contrato de alojamiento incorrectos.** Si el personal no capturó datos obligatorios el sistema mostrará el siguiente mensaje: **"Los datos son obligatorios".** Si capturo datos no válidos, el **sistema** mostrará el siguiente mensaje: **"Los datos son incorrectos".** **(E2) El personal no confirmó la cancelación del contrato.** Si el personal no confirma la cancelación del contrato, el sistema seguirá mostrando la lista de contratos. **(E3) Los datos son incorrectos.** Si el número de cliente es incorrecto el sistema muestra el mensaje: **"Número de cliente incorrecto".** Si el número de contrato es incorrecto el sistema muestra el mensaje: **"Número de contrato incorrecto"**.
POSCONDICIONES	No existen.

9.11 Diagramas de secuencia relativos al cliente

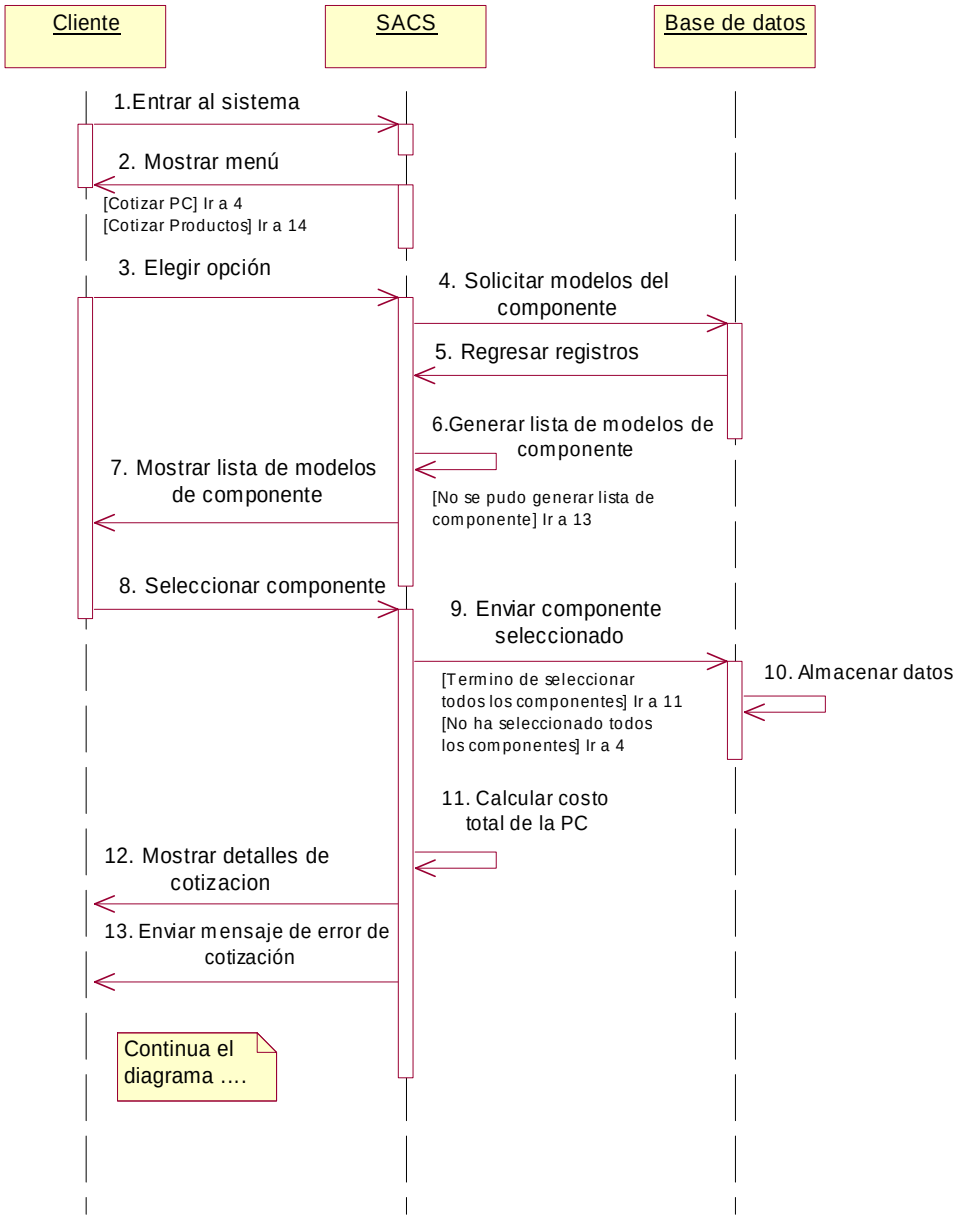
9.11.1 Realizar registro

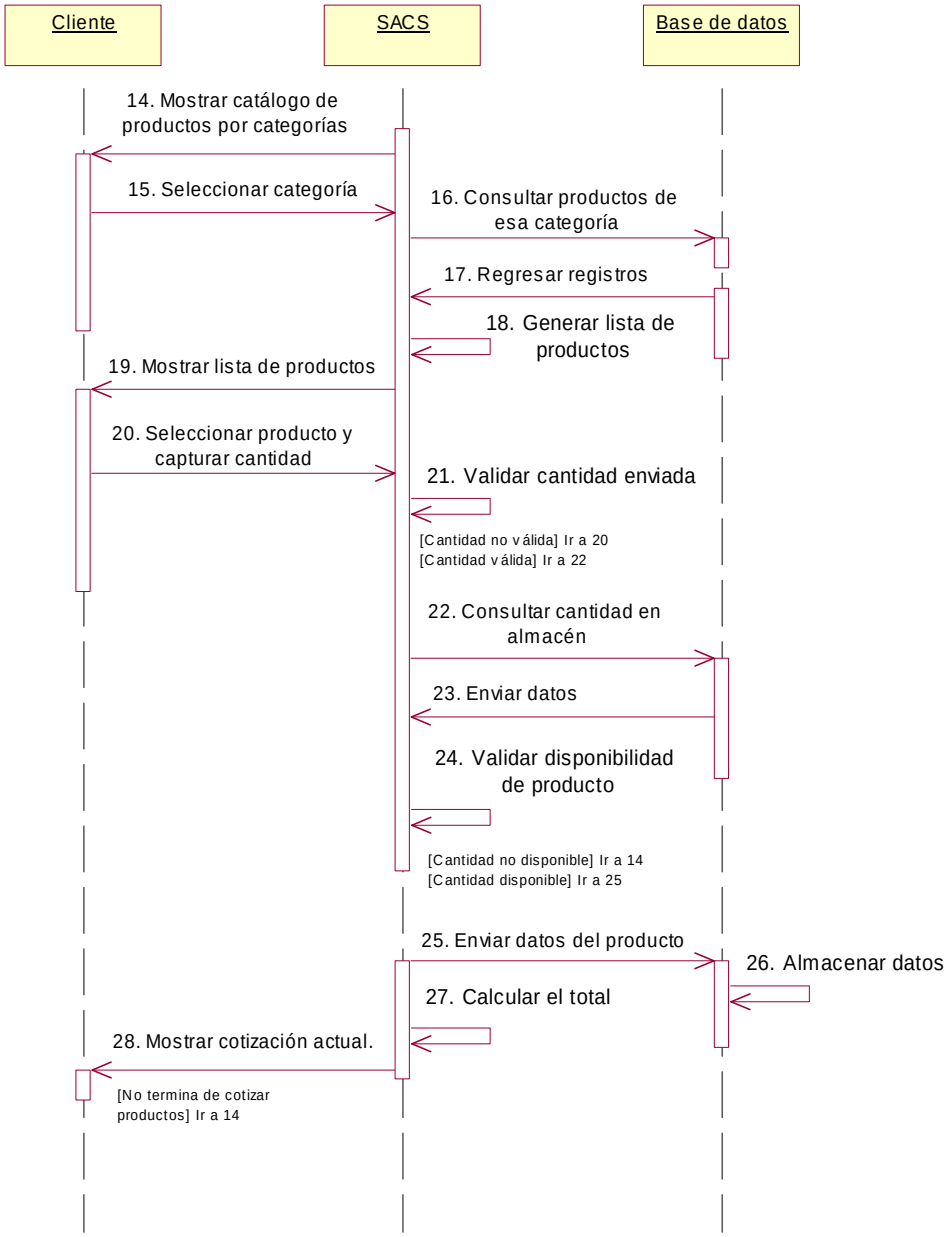


9.11.2 Registrar Autenticación

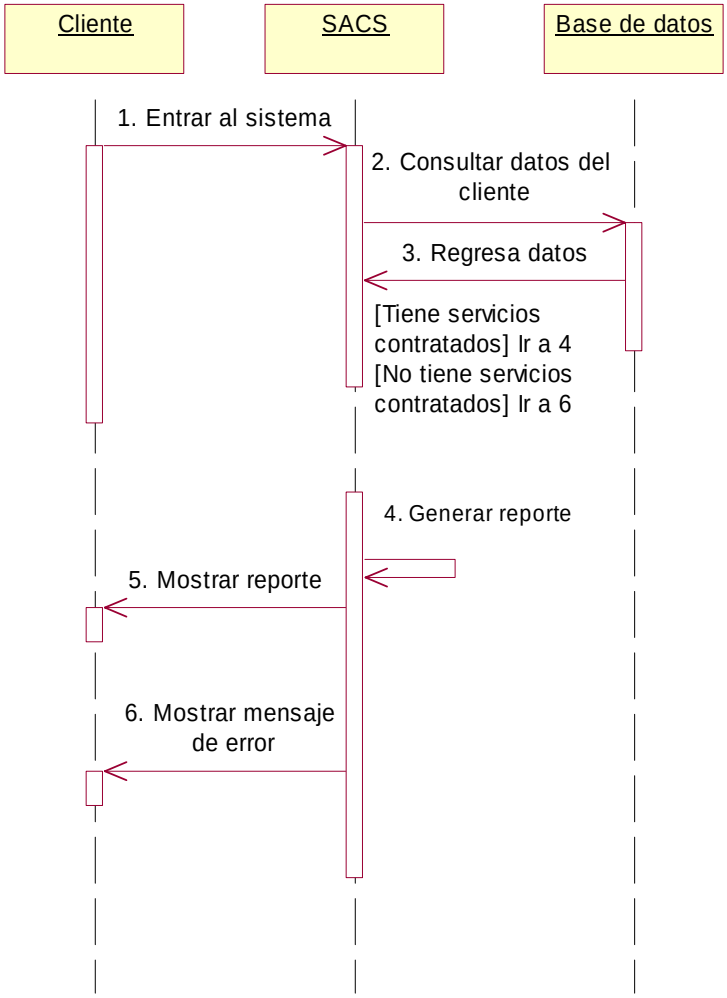


9.11.3 Cotizar productos

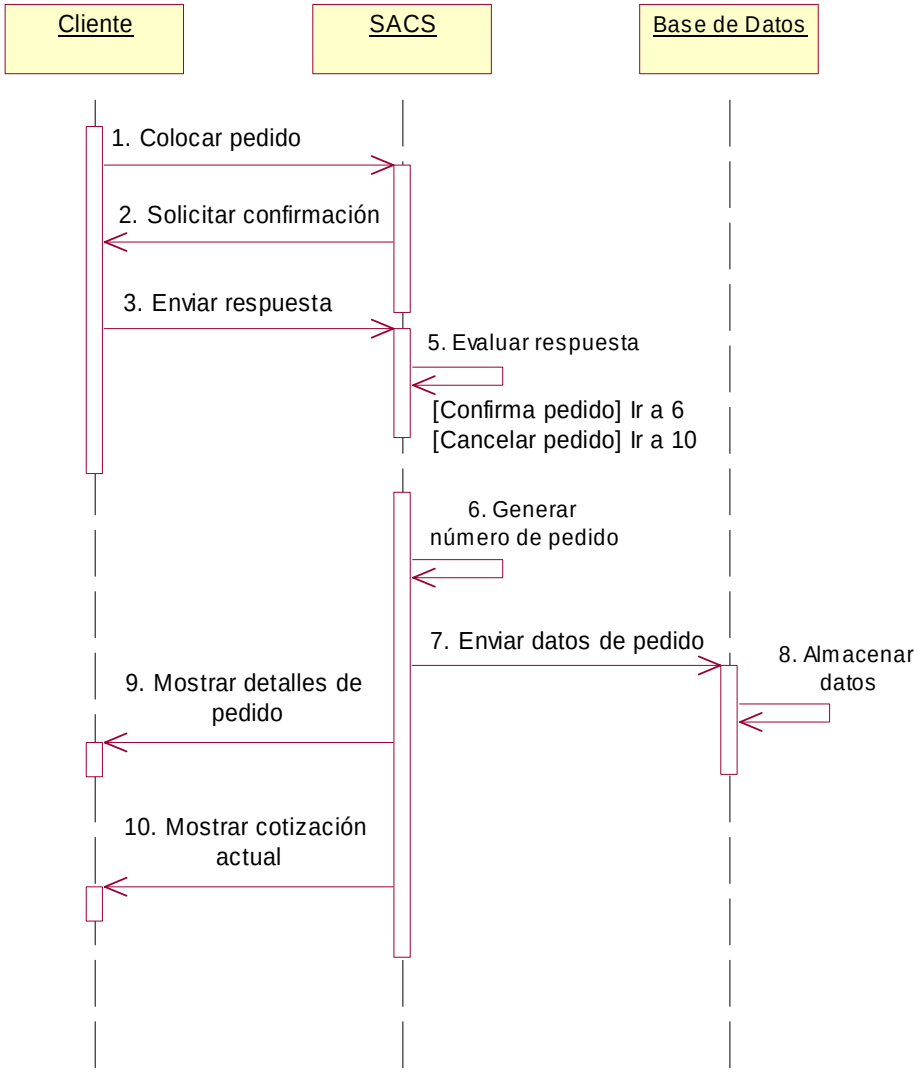



Cliente
SACS
Base de datos
14. Mostrar catálogo de productos por categorías
15. Seleccionar categoría
16. Consultar productos de esa categoría
17. Regresar registros
18. Generar lista de productos
19. Mostrar lista de productos
20. Seleccionar producto y capturar cantidad
21. Validar cantidad enviada
[Cantidad no válida] Ir a 20
[Cantidad válida] Ir a 22
22. Consultar cantidad en almacén
23. Enviar datos
24. Validar disponibilidad de producto
[Cantidad no disponible] Ir a 14
[Cantidad disponible] Ir a 25
25. Enviar datos del producto
26. Almacenar datos
27. Calcular el total
28. Mostrar cotización actual.
[No termina de cotizar productos] Ir a 14

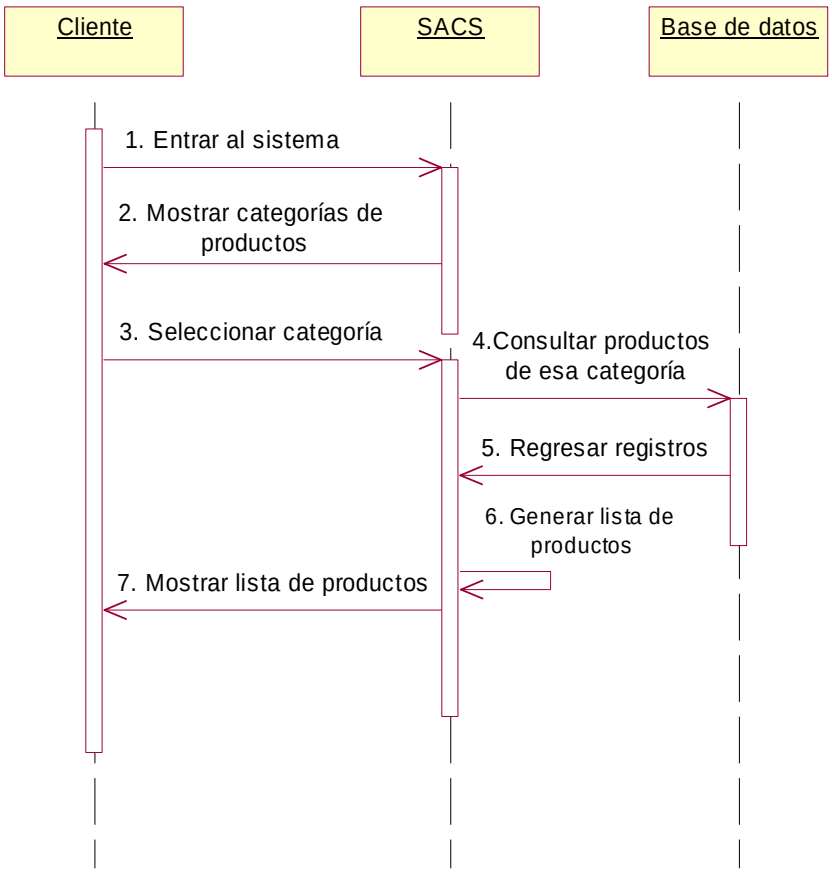
9.11.4 Consultar pagos



9.11.5 Realizar pedido

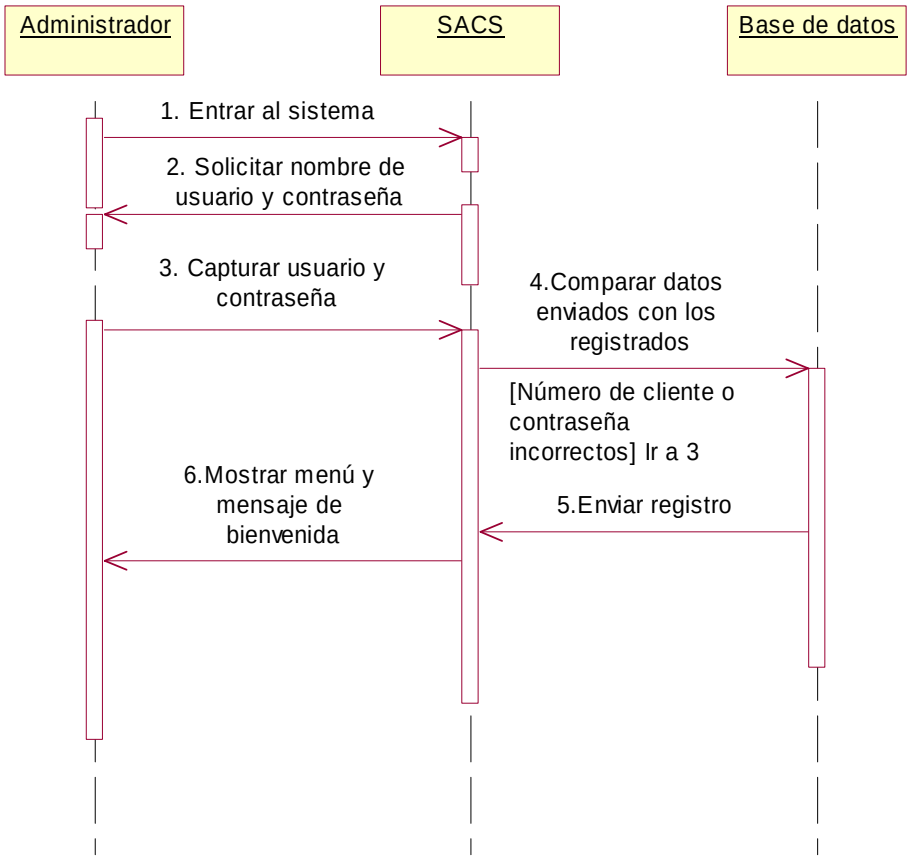


9.11.6 Consultar productos

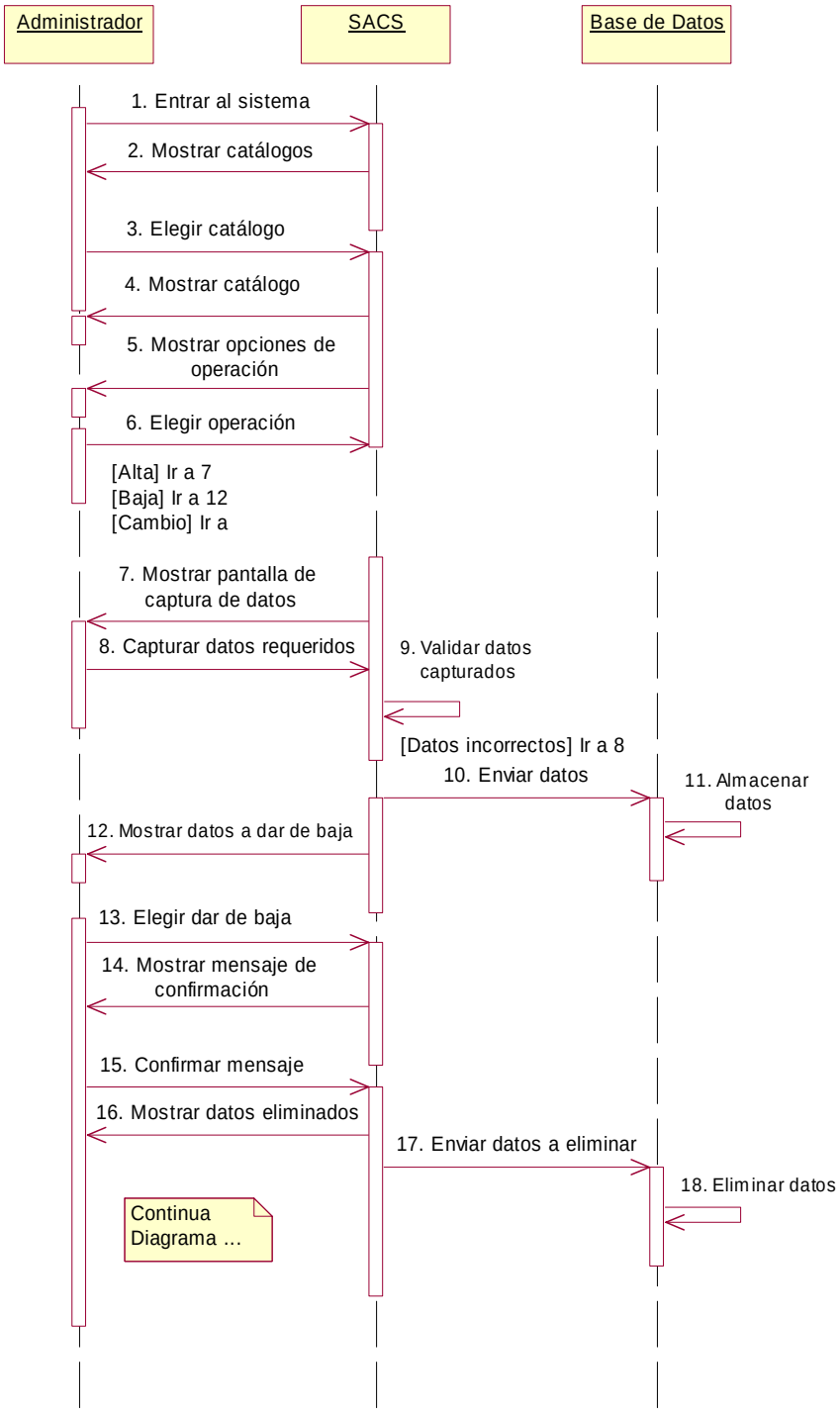


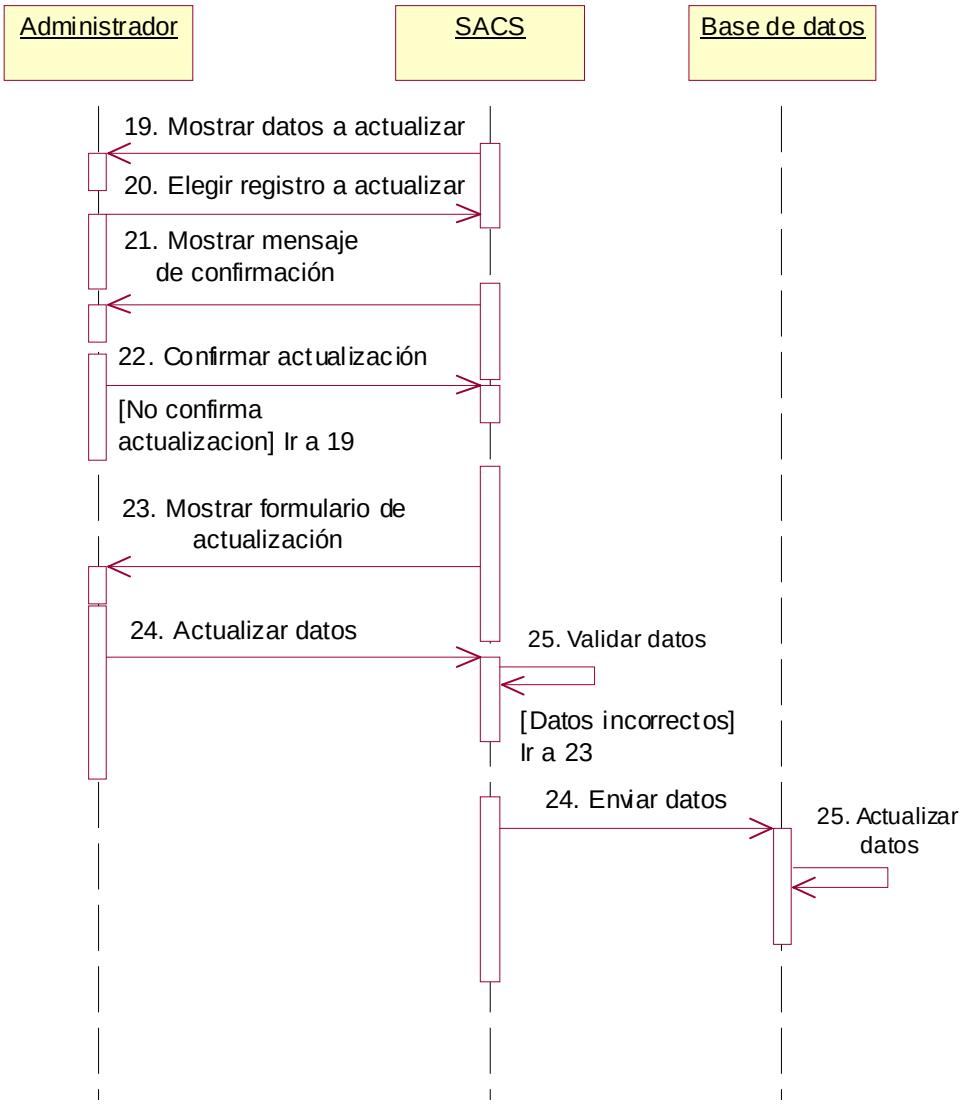
9.12 Diagramas de secuencia relativos al administrador

9.12.1 Registrar autenticación

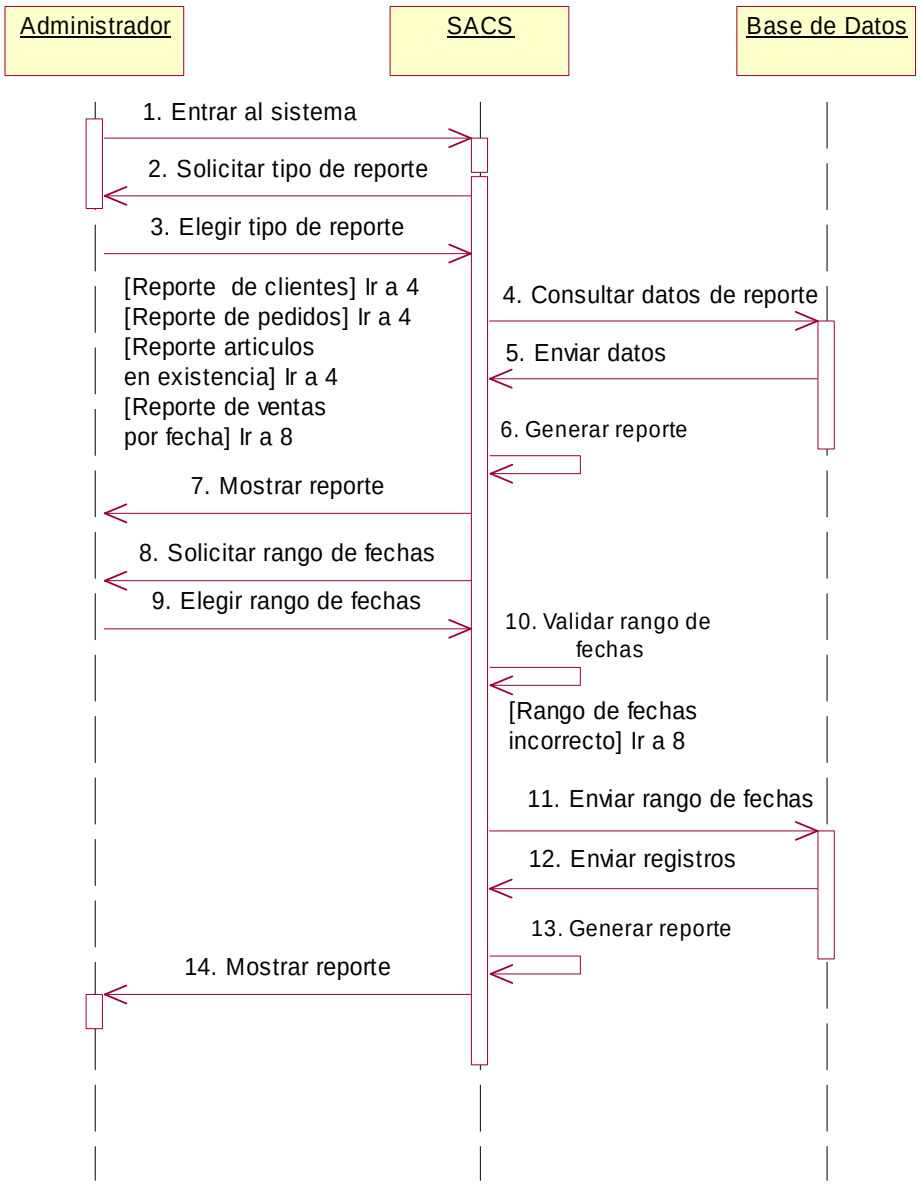


9.12.2 Administrar catálogos




Administrador
SACS
Base de datos
19. Mostrar datos a actualizar
20. Elegir registro a actualizar
21. Mostrar mensaje de confirmación
22. Confirmar actualización
[No confirma actualizacion] Ir a 19
23. Mostrar formulario de actualización
24. Actualizar datos
25. Validar datos
[Datos incorrectos] Ir a 23
24. Enviar datos
25. Actualizar datos

9.12.3 Generar reportes

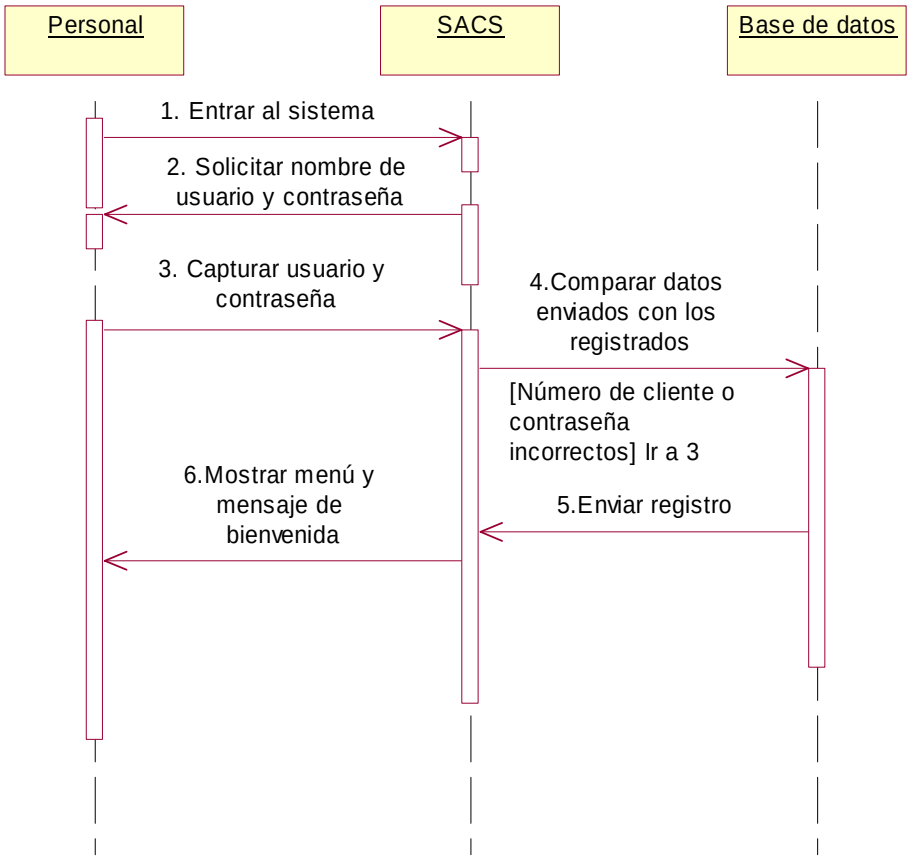


9.12.4 Administrar contrato de alojamiento

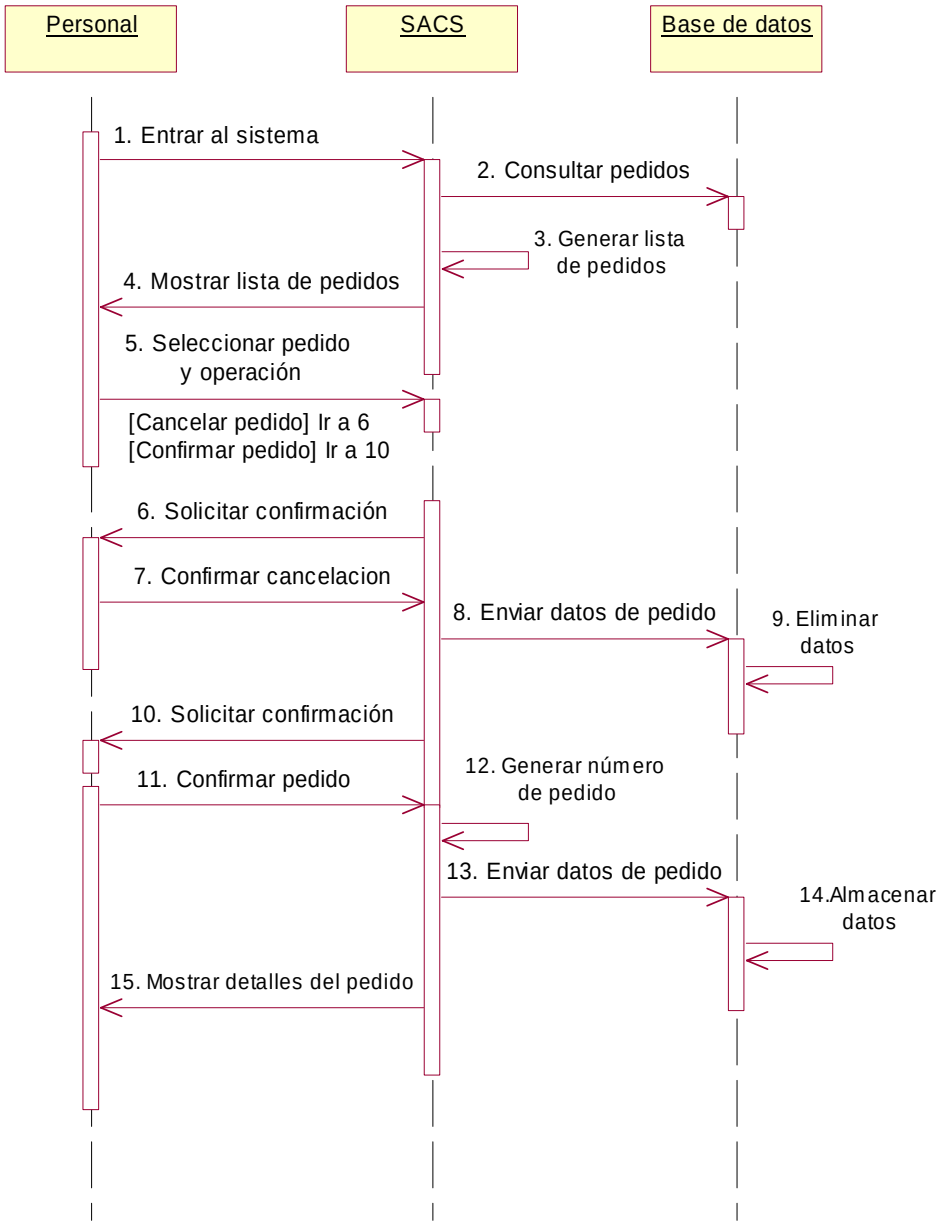
Ver diagrama de secuencia “Administrar contrato de alojamiento” relativo al personal.

9.13 Diagramas de secuencia relativos al personal

9.13.1 Registrar autenticación



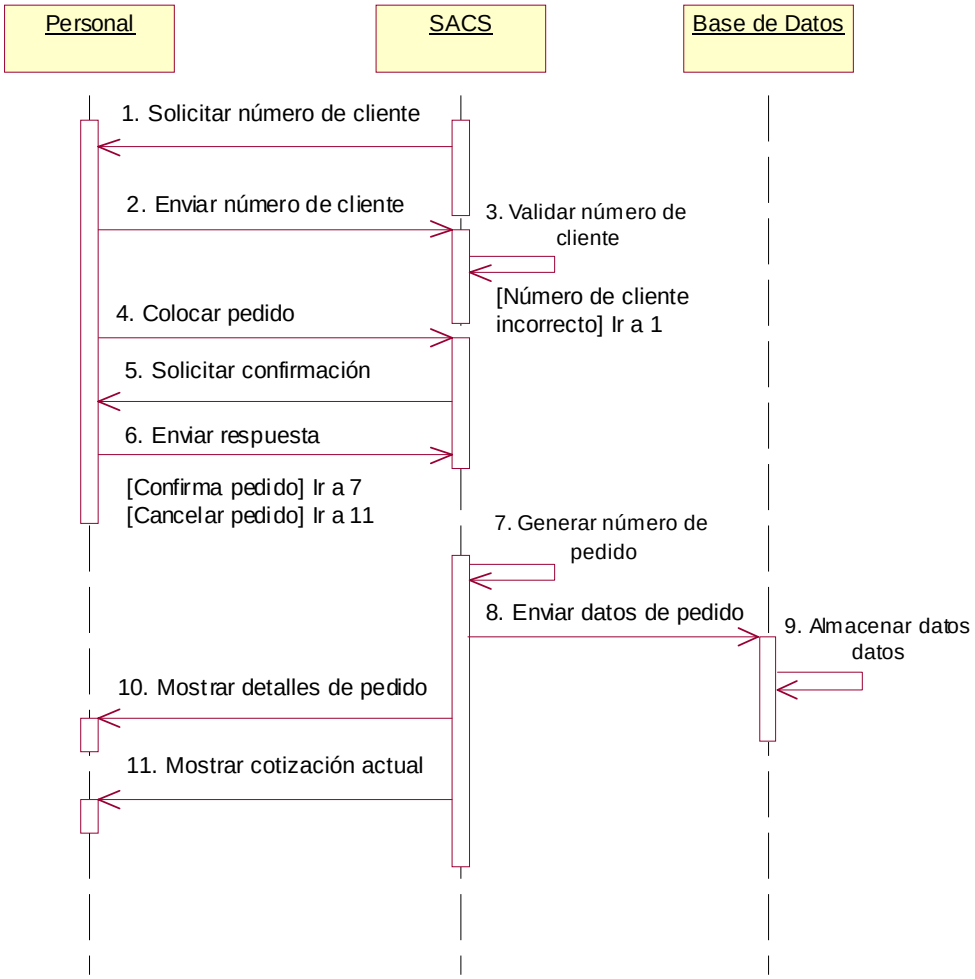
9.13.2 Consultar pedidos



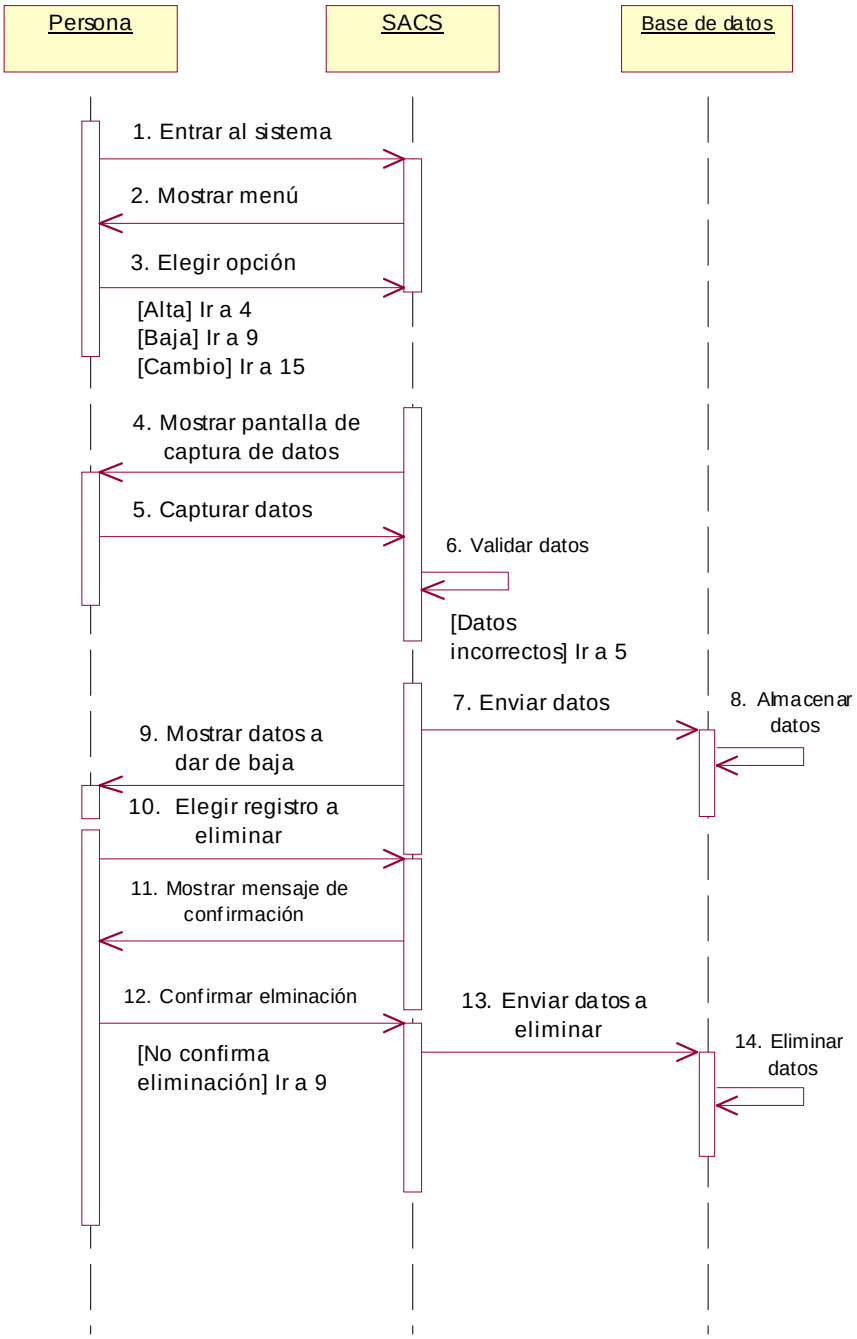
9.13.3 Cotizar Productos

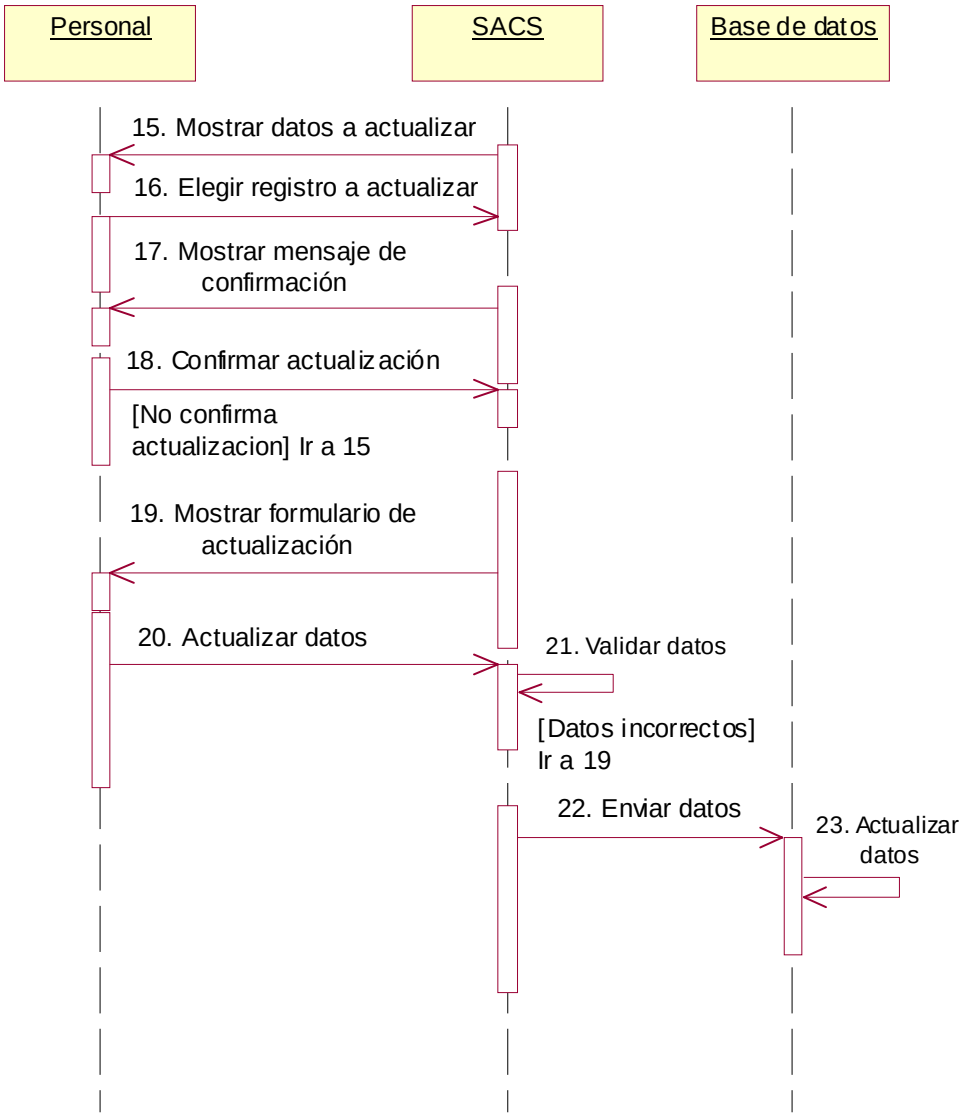
Ver diagrama de secuencia **“Cotizar productos”** relativos al cliente.

9.13.4 Realizar pedido

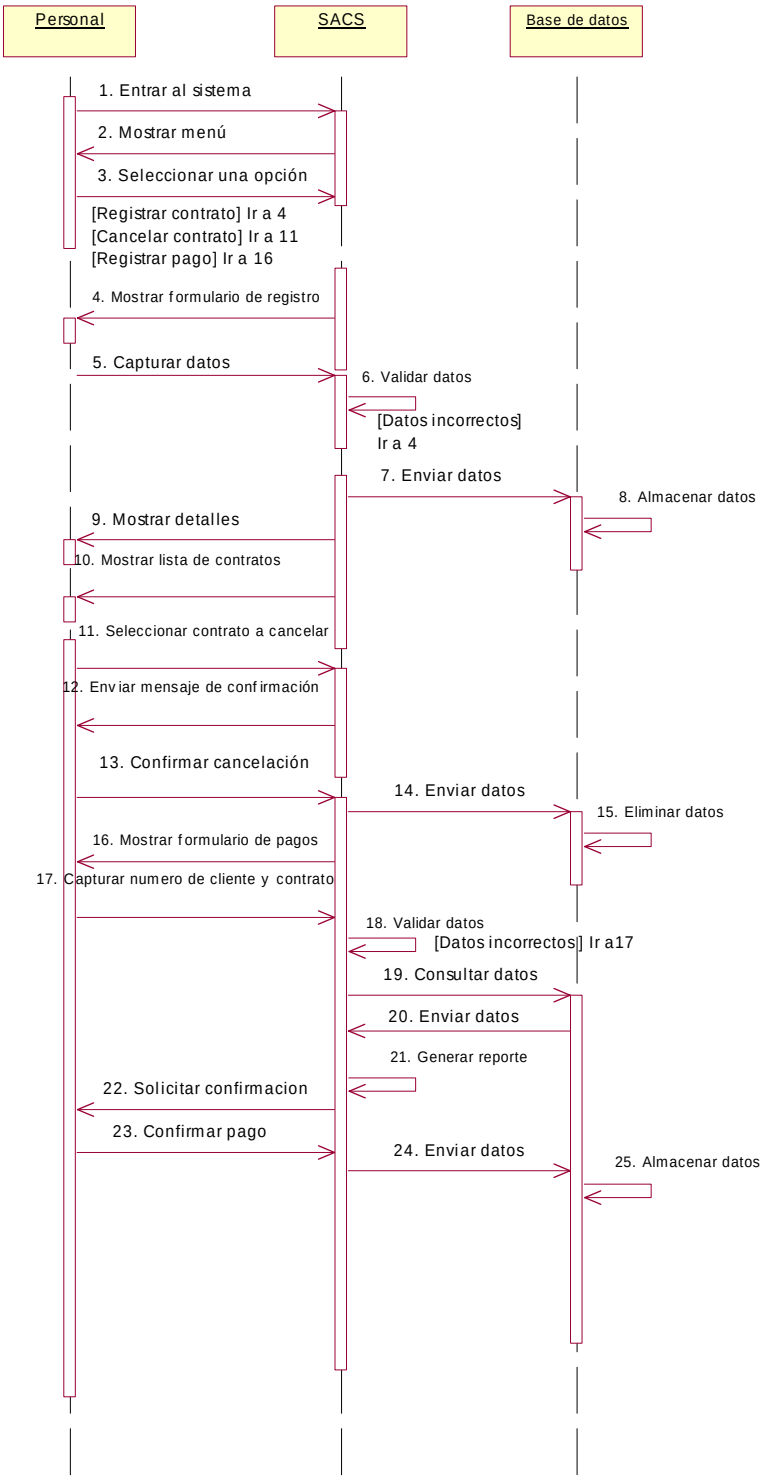


9.13.5 Administrar catálogo de clientes

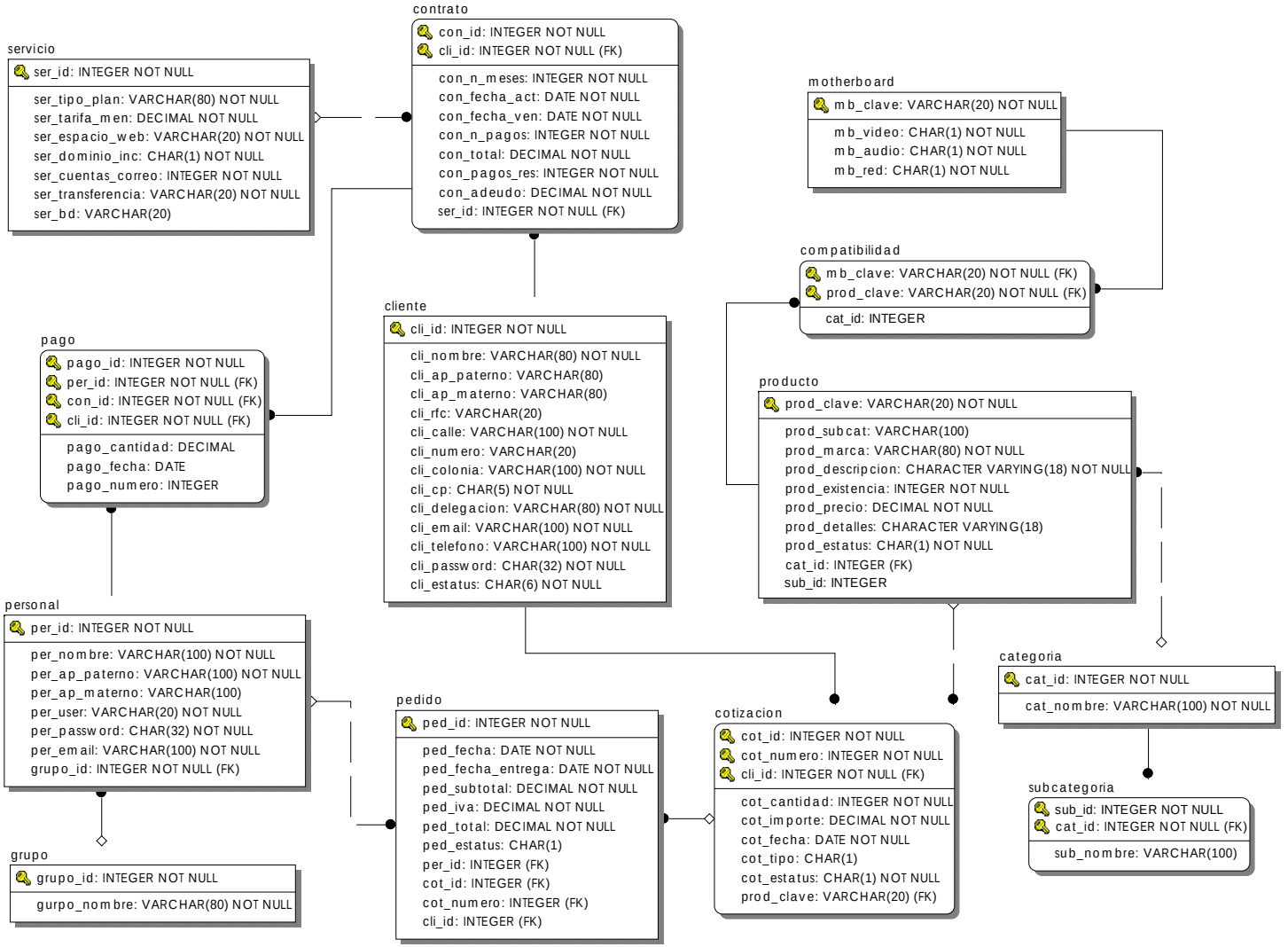



Personal
SACS
Base de datos
15. Mostrar datos a actualizar
16. Elegir registro a actualizar
17. Mostrar mensaje de confirmación
18. Confirmar actualización
[No confirma actualizacion] Ir a 15
19. Mostrar formulario de actualización
20. Actualizar datos
21. Validar datos
[Datos incorrectos] Ir a 19
22. Enviar datos
23. Actualizar datos

9.13.6 Administrar contrato de alojamiento



9.14 Diagrama Entidad Relación



9.15 Diccionario de datos

Tabla: **cliente**						
Almacena los datos del cliente que se registra en el sitio Web de IDEV.						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
cli_id	Número consecutivo de cliente que se asigna cuando alguien se registra en IDEV.	SI	NO	SI	INT2	100
cli_nombre	Nombre(s) del cliente.	NO	NO	SI	VARCHAR(80)	Edgar Ignacio
cli_ap_pat	Apellido paterno del cliente.	NO	NO	NO	VARCHAR(80)	Peña
cli_ap_mat	Apellido paterno del cliente.	NO	NO	NO	VARCHAR(80)	Flores
cli_rfc	RFC del cliente.	NO	NO	NO	VARCHAR(20)	PEFE820731
cli_calle	Nombre de la calle.	NO	NO	SI	VARCHAR(100)	Mariano Escobedo
cli_numero	Número de la calle.	NO	NO	NO	VARCHAR(20)	76 Int.5
cli_colonia	Nombre de la colonia.	NO	NO	SI	VARCHAR(100)	Barrio San Pedro
cli_cp	Código postal.	NO	NO	SI	CHAR(5)	09000
cli_delegacion	Nombre de la delegación.	NO	NO	SI	VARCHAR(80)	Iztapalapa
cli_email	Correo electrónico del cliente.	NO	NO	SI	VARCHAR(100)	edgaripf@yahoo.com.mx
cli_telefono	Teléfono del cliente.	NO	NO	SI	VARCHAR(100)	56-86-00-41
cli_password	Contraseña del cliente.	NO	NO	SI	CHAR(32)	4534543t43655y65y6y56yu6y6yjhgr6
cli_estatus	Nombre del Estatus en el que se encuentra el cliente (activo, baja).	NO	NO	SI	VARCHAR(6)	activo

Tabla: **producto**						
Registra los productos del catálogo de IDEV						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
prod_clave	Clave del producto.	SI	NO	SI	VARCHAR(20)	CP2534ITL08
prod_subcat	Subcategoría a la que pertenece el producto.	NO	NO	NO	VARCHAR(100)	PROCESADORES P-4 SKT-478/478
prod_marca	Marca del producto.	NO	NO	SI	VARCHAR(80)	INTEL
prod_descripcion	Breve descripción del producto.	NO	NO	SI	TEXT	INTEL Pr4 478/800-HT 3.4GZ/1 MB CAJA*IPI
prod_existencia	Cantidad existente en el almacén del producto.	NO	NO	SI	INT2	50
prod_precio	Precio en pesos mexicanos del producto.	NO	NO	SI	FLOAT4	3242.47
prod_detalles	Características adicionales del producto.	NO	NO	NO	TEXT	**Velocidad del procesador:** 3.4Gz **Interfase del procesador:** Socket 478 **Clase del procesador:** Pentium 4 **Cache L2 :** 1MB **Bus:** 800MHz **Tecnología Adicional:** HT
prod_estatus	Caracter que indica si el producto continúa en el catálogo o no, puede tomar los valores S o N.	NO	NO	SI	CHAR(1)	S
cat_id	Clave de la categoría a la que pertenece el producto.	NO	SI	SI	INT2	1
sub_id	Clave de la subcategoría a la que pertenece	NO	NO	NO	INT2	1

Tabla: **motherboad**						
Registra los modelos de motherboard a la venta en IDEV.						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
mb_clave	Clave de la motherboard.	SI	NO	SI	VARCHAR(20)	MC4845ITL25
mb_video	Este caracter indica si la motherboard tiene video incluido o no, los valores que puede tomar son S o N.	NO	NO	SI	CHAR(1)	**S**
mb_audio	Este caracter indica si la motherboard tiene audio incluido o no, los valores que puede tomar son S o N.	NO	NO	SI	CHAR(1)	**S**
mb_red	Este caracter indica si la motherboard tiene tarjeta de red incluida o no, los valores que puede tomar son S o N.	NO	NO	SI	CHAR(1)	**N**

Tabla: **compatibilidad**						
Registra la compatibilidad entre la motherboard y los productos						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
mb_clave	Clave de la motherboard.	SI	SI	SI	VARCHAR(20)	MC4845ITL25
prod_clave	Clave del producto con el cual es compatible la motherboard.	SI	SI	SI	VARCHAR(20)	CP2534ITL08
cat_id	Clave de la categoría a la que pertenece el producto con el que es compatible.	NO	NO	NO	INT2	1

Tabla: **categoría**						
Registra las diferentes categorías de productos						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
cat_id	Clave de la categoría.	SI	NO	SI	INT2	1
cat_nombre	Nombre de la categoría.	SI	SI	SI	VARCHAR(100)	procesadores

Tabla: **subcategoría**						
Registra las diferentes categorías de productos						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
sub_id	Clave de la subcategoría.	SI	NO	SI	INT2	1
sub_nombre	Nombre de la subcategoría.	SI	SI	SI	VARCHAR(100)	PROCESADORES P4 SKT-478/478A
cat_id	Clave de la categoría a la que pertenece.	SI	NO	SI	INT2	1

Tabla: **servicio**						
Registra los diferentes planes de servicios de alojamiento que ofrece IDEV						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
ser_id	Clave del servicio.	SI	NO	SI	INT2	1
ser_tipo_plan	Nombre del servicio de alojamiento.	NO	NO	SI	VARCHAR(80)	básico
ser_tarifa_men	Costo en pesos mexicanos de la tarifa mensual del plan de alojamiento.	NO	NO	SI	FLOAT4	150.00
ser_espacio_web	Espacio de almacenamiento ofrecido.	NO	NO	SI	VARCHAR(20)	150 MB
ser_dominio_inc	Caracter que indica si el	NO	NO	SI	CHAR(1)	N

	servicio incluye el dominio o no, los valores que puede tomar son S o N.					
ser_cuentas_correo	Cantidad de cuentas de correo POP3 que incluye el servicio	NO	NO	SI	INT2	10
ser_transferencia	Transferencia que ofrece el servicio.	NO	NO	SI	VARCHAR(20)	3.00 GB
ser_bd	Texto que indica el soporte para base de datos que incluye el servicio.	NO	NO	SI	VARCHAR(20)	SOPORTE MYSQL

Tabla: **personal**

Registra el catálogo de usuarios del sistema.

Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
per_id	Clave del personal que utiliza el sistema.	SI	NO	SI	INT2	10
per_nombre	Nombre del personal	NO	NO	SI	VARCHAR(100)	Raúl Milton
per_ap_paterno	Apellido paterno del personal.	NO	NO	SI	VARCHAR(100)	Rubio
per_ap_materno	Apellido materno del personal.	NO	NO	NO	VARCHAR(100)	Lorenzana
per_user	Nombre de usuario en el sistema.	NO	NO	SI	VARCHAR(20)	Milton
per_password	Contraseña del usuario.	NO	NO	SI	CHAR(32)	4534543t43655y65y6y56yu6y6yjhgr6
per_email	Correo electrónico del personal.	NO	NO	SI	VARCHAR(100)	milton@idev.com.mx
grupo_id	Clave del grupo al que pertenece el usuario.	NO	SI	SI	INT2	2

Tabla: **grupo**						
Registra el catálogo de grupos de usuarios						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
grupo_id	Clave del grupo	SI	NO	SI	INT2	2
grupo_nombre	Nombre del grupo	SI	SI	SI	VARCHAR(100)	personal

Tabla: **cotizacion**						
Registra los productos que un usuario va agregando a su cotización.						
Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
cot_id	Clave de la cotización	SI	NO	SI	INT2	5
cli_id	Número de cliente al que pertenece la cotización	SI	SI	SI	INT2	100
cot_cantidad	Cantidad del producto agregado a la cotización.	NO	NO	SI	INT	2
cot_importe	Importe calculado con el precio del artículo agregado y la cantidad.	NO	NO	SI	FLOAT4	6484.94
cot_fecha	Fecha de registro del artículo agregado a la cotización.	NO	NO	SI	DATE	2006-01-29
cot_tipo	Caracter que permite saber si el producto agregado corresponde a una cotización de una PC o de productos en general. Los valores que puede tomar son G o P.	NO	NO	NO	CHAR(1)	G
cot_estatus	Caracter que permite saber si el producto sigue en la cotizacion o si ya fue colocado en pedido. Los	NO	NO	SI	CHAR(1)	C

	valores que puede tomar son C o P.					
prod_clave	Clave del producto que fue agregado a la cotización	NO	SI	SI	VARCHAR(20)	CP2534ITL08

Tabla: **pedido**

Registra los pedidos realizados por los usuarios.

Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
ped_id	Clave del pedido.	SI	NO	SI	INT2	150
cot_id	Clave de la cotización que genera el pedido.	SI	SI	SI	INT2	5
cli_id	Número de cliente que envía el pedido.	SI	SI	SI	INT2	100
per_id	Clave del personal que atiende el pedido.	SI	SI	SI	INT2	10
ped_fecha	Fecha en la que se envió el pedido.	NO	NO	SI	DATE	2006-01-30
ped_fecha_entrega	Fecha en la que se entregará el pedido.	NO	NO	SI	DATE	2006-02-5
ped_subtotal	Subtotal generado con la sumatoria del importe de cada artículo agregado a la cotización.	NO	NO	SI	FLOAT4	6484.94
ped_iva	Cantidad de IVA.	NO	NO	SI	FLOAT4	972.74
ped_total	Cantidad total a pagar.	NO	NO	SI	FLOAT4	7,457.68
ped_estatus	Caracter que permite saber el estatus en el que se encuentra el pedido, puede tomar dos valores C (concretado) o X (cancelado)	NO	NO	NO	CHAR(1)	C

Tabla: **contrato**

Registra los servicios de alojamiento contratados por los clientes

Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
con_id	Clave del contrato.	SI	NO	SI	INT2	9
cli_id	Número de cliente que contrata un plan de alojamiento.	SI	SI	SI	INT2	100
con_n_meses	Número de meses de servicio contratados.	NO	NO	SI	INT2	6
con_fecha_act	Fecha de activación del servicio.	NO	NO	SI	DATE	2006-02-01
con_fecha_ven	Fecha de vencimiento del servicio.	NO	NO	SI	DATE	2006-08-01
con_n_pagos	Número de pagos que se tienen que cubrir.	NO	NO	SI	INT2	3
con_total	Cantidad total a pagar.	NO	NO	SI	FLOAT4	900.00
con_pagos_res	Número de pagos que faltan por realizar.	NO	NO	SI	INT2	2
con_adeudo	Cantidad que falta por pagar.	NO	NO	SI	FLOAT4	600.00
ser_id	Clave del servicio contratado.	NO	SI	SI	INT2	1

Tabla: **pago**

Registra los pagos realizados por los usuarios

Nombre del atributo	Descripción	PK	FK	NN	Tipo de dato	Ejemplo
pago_id	Clave del pago.	SI	NO	SI	INT2	10
per_id	Clave del personal que registra el pago.	SI	SI	SI	INT2	10
con_id	Número de contrato al que	SI	SI	SI	INT2	9

	corresponde el pago.					
cli_id	Número de cliente que contrató el servicio.	SI	SI	SI	INT2	100
pago_cantidad	Cantidad del pago.	NO	NO	SI	FLOAT4	300.00
pago_fecha	Fecha en la que se registra el pago.	NO	NO	SI	DATE	2006-02-01
pago_numero	Número de pago.	NO	NO	SI	INT2	1

Conclusión

El concepto de software libre logró adaptarse y evolucionar para competir a la par del modelo propietario. Es de esperarse que esta concepción continúe creciendo rápidamente.

Aún así, en estos momentos todavía este software no ha alcanzado el grado de madurez que le permita desplazar de los escritorios al software comercial, fácil de instalar y utilizar.

Pero así como se reconoce que el software comercial es la solución que más se ajusta a las necesidades del usuario hogareño que solo quiere leer sus mensajes de correo electrónico y jugar video juegos, hay que destacar que en otros ámbitos el software libre es el que va a la cabeza. El mercado de servidores Web es uno de los ejemplos más claros. Apache y GNU/Linux gobiernan Internet, en gran parte por su precio (nulo), y las ventajas que nos ofrecen.

La posibilidad de inspeccionar y modificar códigos fuente, proporciona ventajas interesantes que deberían ser aprovechados en distintos campos, por ejemplo la posibilidad de modificar y redistribuir las herramientas, permite que se personalicen las soluciones para las necesidades puntuales de cada empresa. Algo que el software propietario y cerrado no nos proporciona.

Actualmente en una empresa el desarrollo de sistemas se ha vuelto imprescindible. Al implementar sistemas basados en software libre, se logran reducir los costos de desarrollo del mismo ahorrando gran cantidad de dinero en licencias de software propietario, por supuesto obteniendo la misma calidad.

Bibliografía

- **COGGESHALL, JOHN;** *BIBLIA DE PHP 5:* ANAYA MULTIMEDIA.
- **LEBLANC, DEE ANN;** *BIBLIA ADMINISTRACIÓN DE SISTEMAS LINUX*: ANAYA.
- **STEPHEN R. SCHACH;** *ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS CON UML Y EL PROCESO UNIFICADO*: MC GRAWHILL, 2005.
- **ISAACS, SCOTT;** *A FONDO DYNAMIC HTML:* MC GRAWHILL.
- **GLASS, MICHAEL K.**; *FUNDAMENTOS DESARROLLO WEB CON PHP APACHE Y MYSQL*: ANAYA.
- **KABIR, MOHAMMED J.;** *BIBLIA SERVIDOR APACHE 2*: ANAYA.
- **MEDINETS, DAVID**; *HERRAMIENTAS DE PROGRAMACION SHELL DE UNIX*; MC GRAWHILL.
- **SCHMULLER, JOSEPH;** *APRENDIENDO UML EN 24 HORAS*: PEARSON.
- **ANAYA ÁLVAREZ, CARLOS;** *MÁXIMA SEGURIDAD EN INTERNET*, MADRID: ANAYA MULTIMEDIA, 1998.
- **MAZLAKOWSKI;** *APRENDIENDO MYSQL EN 21 DIAS*: PEARSON.
- **ZIEGLER, ROBERT;** *GUÍA AVANZADA FIREWALLS LINUX*, TRADUCCIÓN JOSÉ IGNACIO SÁNCHEZ, EVA MARÍA LÓPEZ, MADRID, ALFAOMEGA, 2000.
- **APUNTES DE "ADMINISTRACIÓN DE SERVIDORES WWW CON LINUX",** Diplomado "Desarrollo e Implementación de Sistemas con Software Libre y Linux".

- http://httpd.apache.org/docs/2.0/es/mod/directives.html
- http://www.php.net/manual/es/
- http://www.postgresql.org/docs/8.1/interactive/index.html
- http://dev.mysql.com/doc/refman/5.0/en/index.html