UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA SU CONSULTA A TRAVÉS DEL WEB"

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

QUE PARA OBTENER EL TITULO DE INGENIERO EN COMPUTACIÓN

P R E S E N T A

Andrés David García Gómez

A S E S O R

Biol. Lizbeth Heras Lara

MÉXICO 2007

Dedico este trabajo a mis padres:

José Guadalupe García Calderón
Guadalupe Gómez Calderón

Y a mi abuelita:

María Calderón Mendoza

AGRADECIMIENTOS

Siendo este el documento más importante que he realizado, considero necesario agradecer a todas aquellas personas que con su apoyo han hecho que hoy pueda presentarlo.

Es cierto que no soy una persona que tenga mucha suerte pero tengo la fortuna de estar rodeado de gente muy valiosa que me ha enseñado, orientado y estimulado para tener una vida llena de logros y reconocimientos.

Seguramente, y pido una disculpa por ello, omitiré a mucha gente que en el momento de redactar estos párrafos no vienen a mi mente, pero trataré de que sean los menos.

Un titulo profesional no es algo sencillo, se pasa por muchas situaciones que son difíciles de afrontar. En mi caso, tengo una base muy sólida representada por mis padres a quienes les debo todo, son mis maestros, mis amigos, las personas a quien más admiro, pase lo que pase siempre han estado conmigo incondicionalmente. Gracias.

Enseguida tengo que mencionar a la persona que fue como mi segunda madre, mi abuelita, que me cuidó cuando era muy chico y mis papás tenían que trabajar. Ojalá estuviera aquí.

A todos mis profesores y amigos de la primaria, secundaria, bachillerato y la universidad porque cuando se cambia de escuela siempre empezamos de nuevo, y parece que el pasado no fuera importante pero sí lo es.

Al Sr. Raúl Cano Parra, a Yair García (actualmente trabajando en el Club de futbol Pachuca) y a Carlos Paredes Trejo, mis ex-entrenadores que me enseñaron a vivir bajo presión, a tener compromiso hacia las cosas que hemos decidido y que la única forma de tener éxito es a través de cuatro puntos fundamentales, disciplina, consistencia, agradecimiento y motivación.

Ya que estoy hablando de estas cuestiones, también quiero agradecer a todos y cada uno de los compañeros con los que alguna vez he compartido una cancha de futbol porque a través de este juego se gana un sentido de compañerismo y trabajo en equipo que difícilmente se da en otros lugares y después puedes aplicarlo en tu vida.

Cuando se habla de una carrera lo que más recordamos, por obvias razones, es la institución en la que realizas tus estudios superiores. Ahora hablaré de ella, pero primero quiero nombrar de manera muy especial a la Escuela Superior de Cómputo (ESCOM), a la Escuela Superior de Física y Matemáticas (ESFM) y a los amigos que pude hacer en estas instituciones en las que asistía de manera continua a algunas clases. En este lugar conocí a la segunda persona que desearía que estuviera todavía conmigo y que por circunstancias de la vida no está: Virginia. De cualquier forma le agradezco por el tiempo que estuvo a mi lado y no pierdo la esperanza de volver a verla.

De la FES Aragón puedo decir que pasé gran parte de la carrera pensando que debí haber estudiado otra cosa, hoy reconozco que en ella aprendí y realice muchas cosas que ahora aplico y que además me gustan, como la programación y analizar la forma en que funcionan algunas empresas, las actividades extractase, los excelentes torneos de dominó, etc.

Con todo esto va implícita la convivencia con los compañeros, siempre con algunos pasas más tiempo que con otros y me gustaría compartir esto con los más cercanos: Filiberto “el cuñado”, Gerardo “el poblano”, Gustavo “el orejón”, Guillermo “la mema galván”, José “el pepe”, Jonathan “el jonas”, Carlos “la pimpi”, Gerardo “amigo casi hermano”, Israel “yescas”, Ricardo “el mota”, Martín “el morales”, Sergio “el tigre del norte”, Jose manuel “el chivo”, todos los “homeros” también conocidos como “los imperdonables” y todos los “nacidos para perder”.

Mis consentidas: Alma, Fabiola, Jessica, Karla, Lesly Carolina, Alma Basilio, Ruth Mancilla, Dulce María, Rosario, Gaby, Monica, Claudia, Mary y Lety.

Espero que no se me hayan olvidado mucho(a)s.

Y muy en especial quiero agradecer y compartir esto con aquellas que han sido y son muy importantes y determinantes en mi crecimiento como persona: Giselle Cortés, la pequeña Lizbeth Leija, Angélica Martínez, Lidia Arista, Andrea Lizbeth Anaya y Norma Faustinos. A todas ustedes no sabría cómo pagarles lo que han hecho por mí.

Para poder terminar mi carrera me encontré con el infortunio de que era necesario soportar los dolores de cabeza y los tratamientos que trae consigo una neuralgia del trigémino. Hoy debo dar las gracias a todos los doctores que de alguna manera ayudaron para mi recuperación.

No podía olvidarme de una institución que valoro mucho, me refiero a la Dirección General de Servicios de Cómputo Académico (DGSCA), en ella he tenido la oportunidad de cursar un Plan de Becarios en Cómputo de Alto Rendimiento, el Diplomado que hoy me da la oportunidad de titularme y algunos cursos más. Adquirí conocimientos que en ningún otro lado hubiese podido, conocí a gente con un alto nivel Académico (Guillermo, Raúl, Héctor Barrón, Elio, Yolanda, Reyna, Silvia, Ismael, Sandra, David y Carmen) que me han enseñado mucho e hice amigos como Karina, Lorena y Ángel.

Lo más importante es que en este lugar conocí a una mujer admirable, muy inteligente y que tuve la fortuna de que aceptara ser mi asesora: Biol. Lizbeth Heras Lara. Quiero decirte que estoy muy agradecido por todo el apoyo que me has brindado y que no pude elegir mejor asesor.

Gracias también a los Jefes de Carrera: Maestro Jesús Díaz Barriga y Maestro Marcelo Pérez Medel, su secretaria “Vicky” quien, por cierto, siempre nos ayuda a realizar los tramites muy rápido, a mis otros revisores, Ing. Silvia Vega Muytoy, M. en I. Elio Vega Munguía, M. en C. Jesús Hernández Cabrera y en general a la UNAM.

Por la ayuda y sugerencias a mi trabajo agradezco a Carolina Collepardo, Yamna Adriana Vázquez, Rosmeri Ortega, Verónica Barranco, al maestro Enrique García Guzmán y a mi hermana Blanca Leticia.

Gracias de antemano a todos los que me acompañan en mi examen profesional porque es importante para mí.

La ciencia dice que tenemos la capacidad de cambiar de opinión catorce veces en tres décimas de segundo y en ese cambiar me he equivocado muchas veces, lo sé. Aun así y, a pesar de todas las circunstancias, acepto y me gusta la forma en que he vivido y adonde he llegado.

A nadie le gusta imaginar cómo podría ser su vida de haber tomado decisiones diferentes porque normalmente pensamos que podría ser mejor, pero la suerte no cuenta cuando se puede elegir y yo elegí. Tengo un título “Ingeniero en Computación” y lo conseguí con el apoyo de todos ustedes.

David

ÍNDICE

CAPÍTULO IV EL MODELO

CAPÍTULO V LAS VISTAS

CAPÍTULO VI EL CONTROLADOR

CONCLUSIONES DEL PROYECTO

INTRODUCCIÓN

Implantar un sistema de información implica la creación de un equipo de trabajo que evaluará metas y objetivos, preparará a los usuarios para un cambio organizacional y técnico, diseñará y desarrollará la solución mas eficiente y, obviamente, planeará un programa en cuanto a tiempos para satisfacer las necesidades del cliente.

Por **sistema de información** podemos entender lo siguiente: Un conjunto de componentes o procedimientos interrelacionados que obtiene, procesa, almacena y distribuye información para apoyar la toma de decisiones, la coordinación y el control en una organización. También ayudan a los administradores y trabajadores a analizar problemas, visualizar aspectos complejos y crear productos nuevos. La siguiente representación muestra su funcionamiento básico.

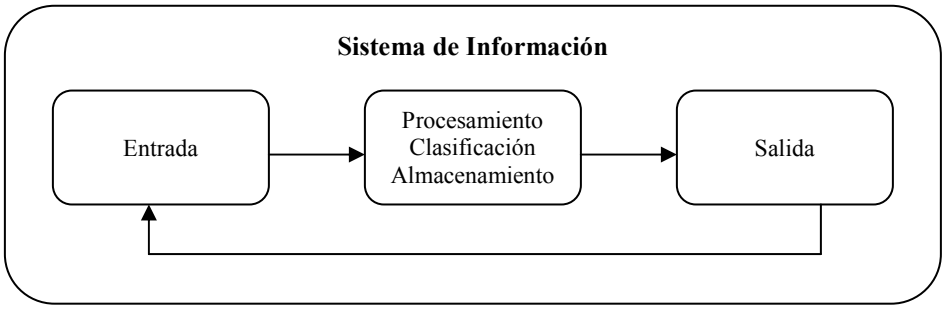


Diagrama básico de un sistema de información.

Los sistemas de información contienen "información" acerca de personas, lugares y cosas importantes dentro de la organización o en su entorno. El término información se refiere a los datos a los que se les ha dado una forma que tiene sentido y es útil para los humanos. Los datos, en cambio, son flujos hechos en bruto que representan sucesos ocurridos en las organizaciones o en el entorno físico, antes de ser organizados y acomodados de tal forma que las personas puedan entenderlos y usarlos.

Etapas para el desarrollo de un sistema de información

1. Investigación Preliminar. Trata de contestar a las preguntas ¿por qué es necesario un proyecto de sistema? y ¿qué se quiere lograr?
2. Determinación de los requerimientos del sistema. Se analizan a fondo los problemas de los sistemas existentes, se identifican los objetivos que debe lograr una solución a esos problemas y se describen soluciones alternas.
3. Diseño del sistema. Produce las especificaciones de diseño lógico y físico para el sistema. Indica los datos de entrada, aquellos que serán calculados y los que deben ser almacenados en una base de datos que servirá a los administradores en la toma de decisiones. Así mismo, se describen con todo detalle los procedimientos de cálculo y los datos individuales. Los diseñadores son los responsables de dar a los programadores las especificaciones de software completas y claramente delineadas.

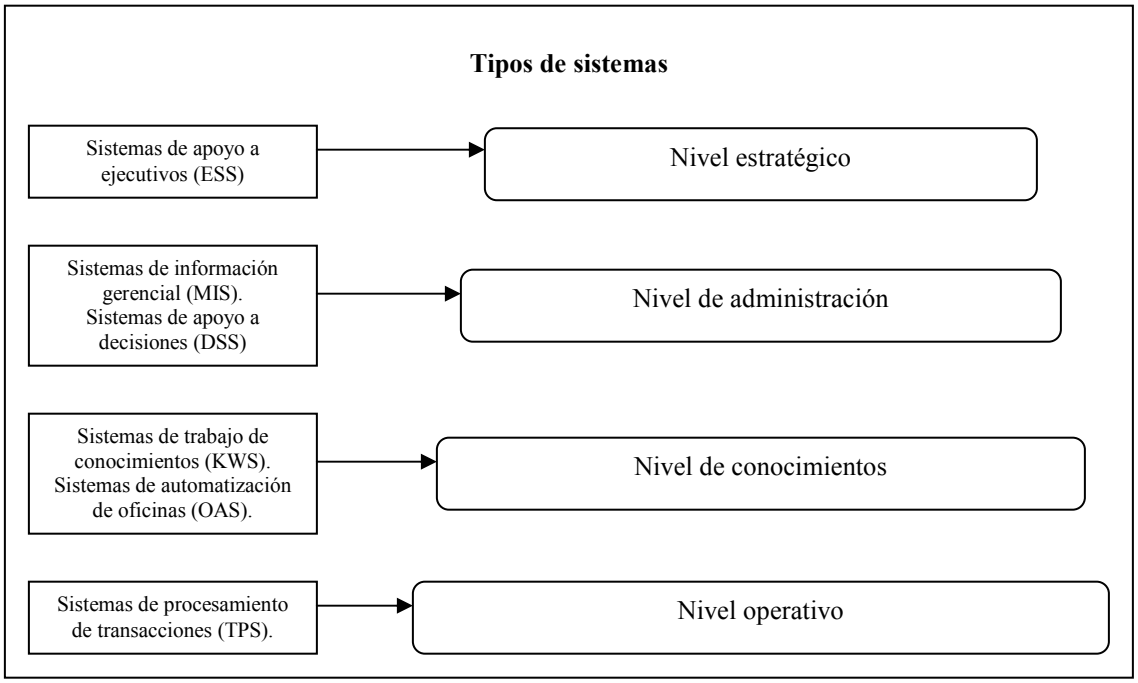
4. Desarrollo de Software. Los encargados de desarrollar software pueden instalar paquetes comprados a terceros, utilizar software libre o escribir programas diseñados a la medida del solicitante. La elección depende del costo de la factibilidad de cada alternativa.
5. Prueba de los sistemas. Consiste en poner a funcionar el sistema nuevo o modificado: pruebas, capacitación y conversión.
6. Implantación. Uso, evaluación y mantenimiento del sistema una vez que se ha instalado y está en producción.

Características generales de un sistema de información

- Contener información interna y externa a la organización.
- Confiabilidad y eficiencia.
- Flexibilidad.
- Seguridad.
- Facilitar la comprensión de la información.
- Consistencia e Integración.
- Que pueda ser utilizado por todos escalones de la estructura jerárquica de la organización.
- Fácil mantenimiento.

Clasificación de los sistemas de información

Una de las formas de clasificar a los sistemas de información es mediante sus objetivos en los diferentes niveles de una organización, como lo muestra el siguiente esquema.



Clasificación de los sistemas de información respecto al nivel organizacional.

OBJETIVOS

OBJETIVO GENERAL: Proporcionar una visión objetiva en cuanto a la implementación de sistemas de información empleando las herramientas de software libre aprendidas en el Diplomado de "Desarrollo e Implementación de Sistemas con Software Libre en LINUX".

OBJETIVOS ESPECIFICOS:

- **CAPÍTULO I**: Plantear las bases del diseño y desarrollo de software de un sistema de información

- **CAPÍTULO II**: Presentar una descripción de las herramientas a utilizar, es decir, una memoria de cada módulo del diplomado. Además de LINUX, HTML, el Servidor Web Apache, PHP, MySQL y PostgreSQL, se incluirán algunas herramientas interesantes como ADODB, JavaScript, Hojas de Estilo(CSS), UML y herramientas CASE.

- **CAPÍTULO III**: Describir los pasos sugeridos para la evaluación de la base de datos, esto para aumentar la confiabilidad y evitar inconsistencias, considerando aspectos técnicos así como los intereses de los involucrados.

- **CAPÍTULO IV**: Identificar las responsabilidades de la lógica de negocio (modelo) y el por qué la conveniencia de separarla de la vista.

- **CAPÍTULO V**: Identificar los puntos importantes al momento de generar la vista adecuada para cada consulta y hacer énfasis en la interactividad y el perfil que debe considerarse con cada usuario en nuestras páginas Web.

- **CAPÍTULO VI**: Detallar las principales tareas de un controlador.

JUSTIFICACIÓN

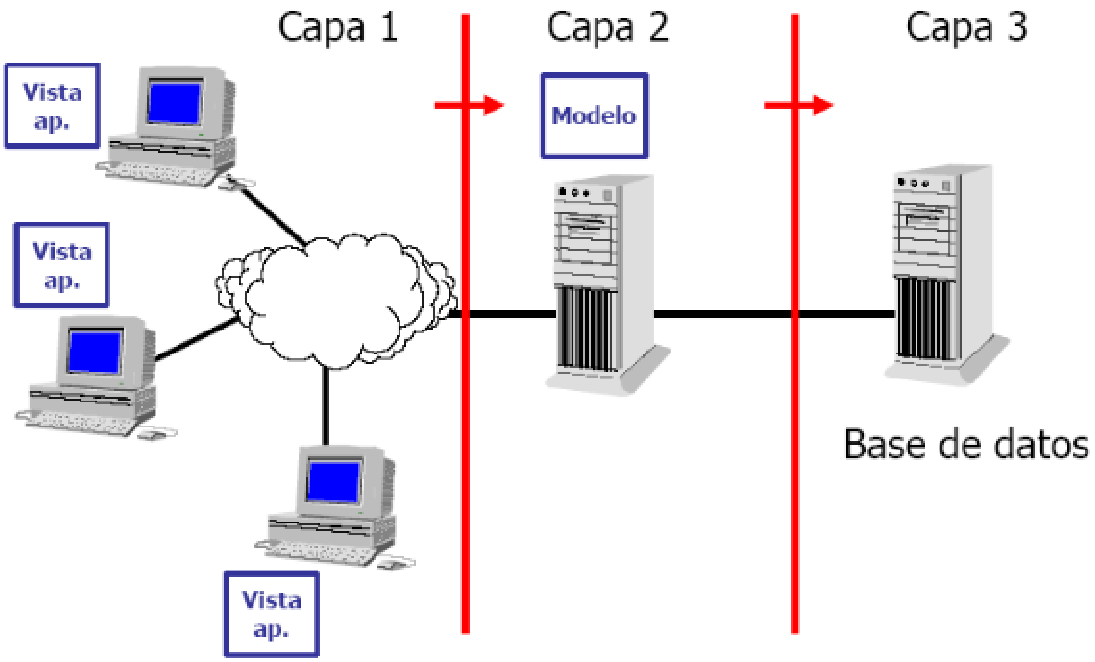
Alcances y límites de la investigación

La metodología de investigación de la tesina presentada aporta una visión general de las etapas de desarrollo de un sistema de información. Despierta la inquietud por profundizar en los temas y, sobre todo, tomando ésta como guía, permite generar o actualizar un sistema de acuerdo a las necesidades del lector. Se tocan la mayoría de los temas contemplados en el plan de estudios del Diplomado en "Desarrollo e Implementación de Sistemas con Software Libre en LINUX".

Como ya mencionamos antes, son seis las etapas básicas para el desarrollo de un sistema de información. En está investigación supondremos que las primeras dos ya se han realizado, es decir, la investigación preliminar y la determinación de los requerimientos del sistema. Por lo tanto, Nos enfocaremos en analizar las etapas de diseño del sistema y desarrollo de software utilizando las herramientas de software libre citadas anteriormente. Por último, y debido a lo corto que debe ser este documento, queda al lector la tarea de documentarse acerca de las etapas de prueba e implantación del sistema.

Aunque algunos temas, como por ejemplo los aspectos específicos de calidad, no se toquen, trataremos de hacer en la medida de lo posible sugerencias o comentarios.

CAPÍTULO I
DISEÑO DEL SISTEMA



Muchos sistemas requieren de tal cantidad de tiempo y dinero para su desarrollo o son tan deficientes en su funcionamiento, que es imposible obtener beneficios de ellos. Ahora bien, lo importante aquí es identificar los motivos que llevan a un mal desarrollo o un mal uso del mismo. En este capítulo mencionaremos las causas principales de este tipo de problemas, así como los factores que hay que tener en cuenta para estar más cerca del éxito.

1.1 Causas más comunes de fracaso en el desarrollo de sistemas

La mayor parte de los problemas que podemos tener los encontramos en el diseño, en los datos, en las operaciones, en la implantación y en el costo.

El diseño real del sistema, desde las primeras etapas (investigación preliminar y determinación de los requerimientos del sistema) puede no haber identificado las necesidades fundamentales o la alternativa adecuada de acuerdo a los recursos con los que se cuenta.

Lo más común es que no se haya recabado la información adecuada (entrevistas, cuestionarios, etc.) acerca del problema por lo que su posterior análisis dará resultados equivocados. Las herramientas elegidas tal vez no son las mas adecuadas o simplemente los estudios de factibilidad no fueron evaluados por todas las personas involucradas en el proceso.

Como desarrollador, uno tiene la obligación de proporcionar varias alternativas de solución con un costo/beneficio viable y además, hacer recomendaciones acerca de lo más conveniente.

En cuanto al funcionamiento del sistema, es importante que funcione con rapidez y que los resultados se muestren en un formato “amigable” para las personas que lo usarán y que sea confiable. Si tienen que interactuar con el sistema, que no sea de una forma complicada y desalentadora, en este sentido adquiere mucha importancia el buen diseño de la interfaz gráfica de usuario (GUI).

Otra causa de errores comunes es que los datos tienen un alto nivel de inexactitud e inconsistencia, por ejemplo, las bases de datos no están debidamente diseñadas o sus campos presentan muchas ambigüedades, alguna parte de la información podría estar inaccesible en determinados momentos por exceso de control (contraseñas o bloqueos mal empleados) etc.

En la parte operacional, generalmente se habla de retraso en la entrega de la información, sobre todo en un sistema que corre en línea. Se debe tener cuidado principalmente con el manejo de sesiones y el número de conexiones a la base de datos que se permiten a la vez. También es recomendable que nuestro código (en este caso PHP) no emplee ciclos o bloques innecesarios, es decir, que sea un código eficiente.

Más allá de lo que pueda causar un mal funcionamiento, es muy importante no olvidar que un sistema tiene que ser compatible, siempre, con la estructura, cultura y metas de la organización que lo está implementando.

Ahora, debemos tener en cuenta que implantar significa realizar un cambio organizacional y técnico, y que casi siempre pasan inadvertidos los problemas significativos del cambio organizacional, de ahí el fracaso de la implantación técnica. El costo relativo de la implantación puede ser alto si se compara con el diseño. La razón de ello es el elevado precio de los paquetes de software, la capacitación, el equipo y suministros, la depuración y preparación de los manuales de operación (documentación).

1.2 Arquitectura de software (El MVC)

Una Arquitectura de software, también denominada **Arquitectura lógica**, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

Una arquitectura de software se selecciona y diseña con base en los objetivos y restricciones. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Las más conocidas son las siguientes:

- **Monolítica**: El software se estructura en grupos funcionales muy acoplados.
- **Cliente-Servidor**: El software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- **Arquitectura de tres niveles**: Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes con un reparto claro de funciones: una capa para la presentación, otra para el cálculo y otra para el almacenamiento.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software (de tres niveles) que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML (y PHP en nuestro caso) y el código que provee de datos dinámicos a la página, el controlador es el sistema de gestión de base de datos y el modelo es la lógica del negocio. Por tanto esto implicara, al tener las capas separadas, un sistema mucho más fácil de mantener y poder reutilizar en el futuro.

Modelo: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos; por ejemplo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos) para almacenar los datos. El MVC no menciona específicamente esta capa de acceso a datos. La capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio.

Aunque se pueden encontrar diferentes implementaciones del MVC, el flujo que sigue el control generalmente es muy parecido al de la figura 1.1 y es el siguiente:

- El usuario interactúa con la interfaz de usuario.
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos.
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario, donde se reflejan los cambios en el modelo, sin embargo, éste no debe tener conocimiento directo sobre la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
- La interfaz espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

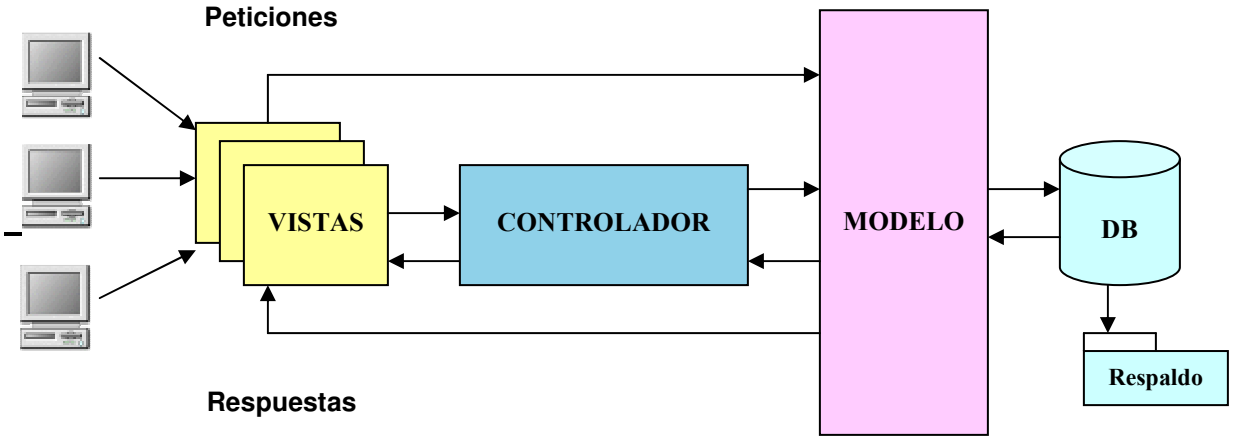


Fig. 1.1. El Modelo Vista Controlador (MVC).

En el desarrollo de software, un **framework** es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otras herramientas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Los más usados para aplicar el MVC con PHP se muestran en la tabla 1.1.

Lenguaje	Licencia	Nombre
PHP	MIT	CakePHP
PHP	GNU/GPL	Kumbia
PHP	MIT	Symfony
PHP	MIT	QCodo
PHP	GNU/GPL	CodeIgniter

Tabla 1.1. Frameworks MVC para PHP.

1.3 Patrón de diseño

Dentro de nuestro modelo, dependiendo de cómo queremos que funcione, es recomendable (no necesario) aplicar un patrón de diseño adecuado a nuestras necesidades. Un **patrón de diseño** es una solución a un problema de diseño no trivial que es efectiva y **reusable**, es decir, se puede aplicar a diferentes problemas de diseño en distintas circunstancias. Se clasifican en tres tipos:

- **De Creación**: Abstraen el proceso de creación de instancias (Abstract Factory, Builder, Factory, Method, Prototype, Singleton).
- **Estructurales:** Se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño (Adapter, Bridge, Composite, Decorador, Facade, Flyweight, Proxy).
- **De Comportamiento:** Tienen que ver con los algoritmos y la asignación de responsabilidades entre objetos (Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor).

1.4 Éxito de los sistemas

En general, los aspectos más importantes para medir el éxito de los sistemas de información son los mostrados en la figura 1.2.

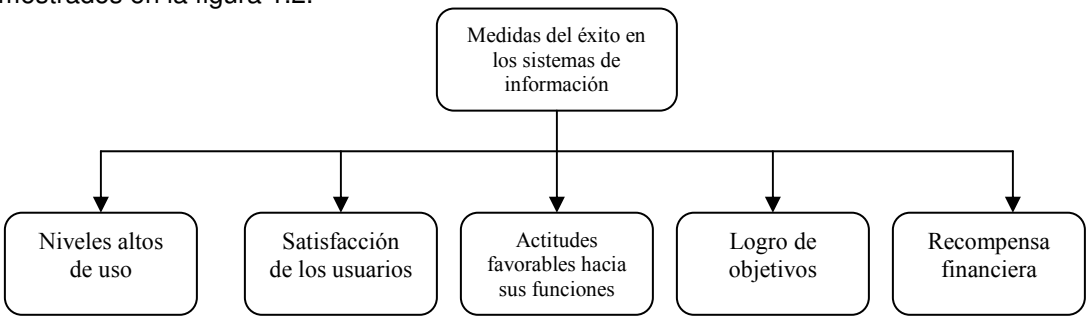


Fig. 1.2. Medidas de éxito.

La **implementación** se refiere a todas las actividades de la organización encaminadas a adoptar, administrar y hacer rutinaria una innovación. El resultado de la implementación puede estar determinado, en gran medida, por el rol de los usuarios, el grado de apoyo administrativo, el nivel de riesgo y complejidad del proyecto y la calidad de manejo del proceso (figura 1.3). Se encuentran indicios del éxito o fracaso en las áreas de diseño, costo, operaciones o datos del sistema de información.

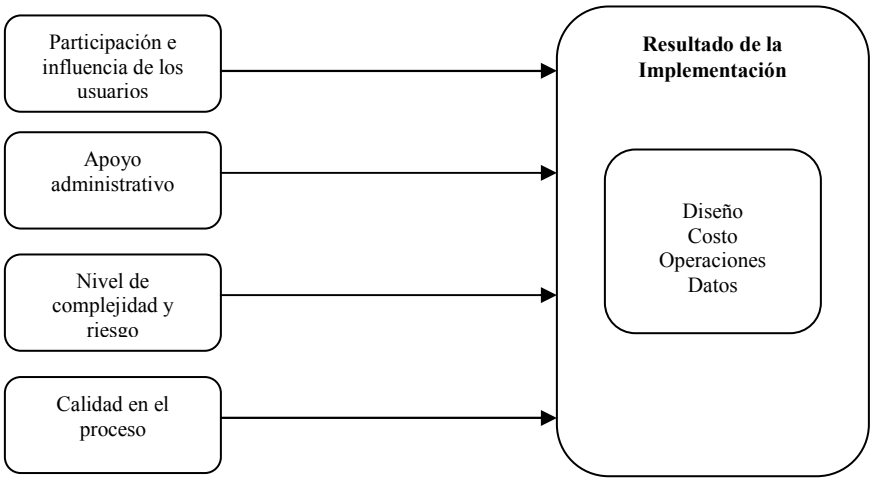


Fig. 1.3. Factores de éxito o fracaso de los sistemas de información.

CAPÍTULO II
LAS HERRAMIENTAS

El software libre es aquel que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. En este apartado describiremos brevemente cada uno de los módulos del diplomado que, en complemento con otras herramientas, hacen posible un buen desarrollo e implementación de un sistema con software libre y que éste sea consultado a través del Web.

En general los temas son: Un sistema operativo (Linux), un servidor Web (Apache), un lenguaje de programación (PHP), el lenguaje de marcas hipertextuales (HTML) para la creación de páginas Web, dos manejadores de bases de datos (PostgreSQL y MySQL), aspectos de seguridad, herramientas CASE, la librería ADODB para darle más portabilidad a nuestra aplicación y algunas otras herramientas interesantes.

2.1 Módulo I. Sistema operativo Linux

Aquí vamos a identificar los métodos de operación y administración básica de Linux, para ello se describen los siguientes puntos:

La historia de Linux inicia con el desarrollo de un pequeño programa llamado Minix[1], un sistema operativo tutorial tipo Unix, escrito por el científico Andrew Tanenbaum[2], este sistema operativo ganó popularidad siendo desarrollado para poder trabajar en diferentes plataformas de equipo de cómputo.

En 1990 un estudiante finlandés inicio un proyecto personal basado en Minix, su intención era desarrollar un sistema compatible con plataforma PC el cual fuera más confiable que Minix, comenzó creando drivers para dispositivos de hardware en lenguaje C, mejorando su plataforma, para 1991 libera la primera versión de Linux sin ser una distribución oficial, la 0.01.

En el mismo año pone a disposición del mundo a través de Internet la primera versión oficial de Linux, la 0.02. Linux, aunque es un sistema libre de licencia GNU, cumple con las normas oficiales de Unix dictadas por POSIX (“Portable Operating System Interface”; la X viene de Unix).

Características de Linux

- Es un sistema operativo de tiempo compartido.
- Multiusuario.
- Multitarea.
- Multiproceso.
- Multiplataforma.
- Modular.
- Portable entre sistemas abiertos.
- Programable.
- Es un sistema tipo cliente-servidor.

Filosofía de Linux

- Su forma modular lleva a su filosofía principal “Entre más pequeño mejor”.
- Permite la reducción de esfuerzo combinando las herramientas.
- Es un sistema personalizable de a cuerdo a las necesidades de cada usuario.

[1] Minix es un clon del sistema operativo Unix distribuido junto con su código fuente.

[2] Director del Departamento de Sistemas de la Universidad de Vrije, Ámsterdam (Países Bajos). Es profesor de Arquitectura de ordenadores y sistemas operativos. Se licenció en el Instituto Tecnológico de Mássachussets, doctorándose en la Universidad de Berkeley.

Su potencia se basa más en la relación entre programas que en los programas mismos.

Plataformas y distribuciones más comunes

El tipo de plataforma se refiere a la arquitectura del hardware sobre el que va a correr nuestro Linux. Las plataformas principales son:

- I386, para equipos tipo PC compatibles.
- PPC, para equipos Maquintoch.
- Sparc, para estaciones de trabajo.
- Alpha, para arquitectura de trabajo con procesadores en paralelo de tipo alpha.

Existen muchas distribuciones, la diferencia radica en que los diseñadores y programadores le dan su toque especial, principalmente en presentación de la interfaz gráfica, implementación de nuevas herramientas, programas, formatos gráficos, etc.

Las distribuciones más comunes son: Slackware, Red Hat, Debian, Mandrake, Suse, Centos, etc. En este diplomado se usa la distribución Slackware (http://www.slackware.com/), veamos algunas características de esta distribución:

Es un avanzado sistema operativo Linux, diseñado con dos objetivos, facilidad para usar y estabilidad como meta prioritaria. Incluyendo el más popular software reciente mientras guarda un sentido de tradición proporcionando simplicidad y facilidad de uso junto al poder y la flexibilidad, Slackware trae lo mejor de todos los Linux a su disposición. Basado en Unix, ahora se beneficia de la contribución de millones de usuarios y desarrolladores alrededor del mundo. Los requerimientos de instalación se muestran en la tabla 2.1.

Slackware Linux proporciona a los nuevos y a los experimentados usuarios por igual un sistema con todas las ventajas, equipado para servidores, puestos de trabajos y máquinas de escritorio. Web, ftp, mail están listos para usarse al salir de la caja, así como una selección de los entornos de escritorio más populares. Herramientas para programación, editores, así como las librerías actuales son incluidas para aquellos usuarios que quieren desarrollar o compilar software adicional.

Hardware	Requiere
Procesador	386
RAM	16 MB – 64M
Espacio en Disco	500MB – 3G
Floppy Drive	1.44 MB
Unidad CD-ROM	
Software	(página oficial)

Tabla 2.1. Requerimientos de instalación.

Componentes del sistema operativo Linux

Linux esta compuesto básicamente de cuatro capas (figura 2.1):

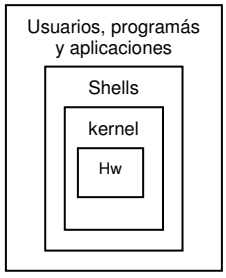


Fig. 2.1. Capas de Linux.

La capa más interna esta conformada por el hardware, que es el conjunto de piezas físicas del equipo de cómputo.

La segunda capa es el Kernel o núcleo del sistema, su función principal es interpretar las instrucciones proporcionadas por el usuario y convertirlas en lenguaje de máquina e indicarle al hardware lo que tiene que realizar con dicha información. Las versiones del Kernel se numeran con 3 números, de la siguiente forma: XX.YY.ZZ.

XX: Indica la serie principal del Kernel. Este número cambia cuando la manera de funcionamiento del Kernel ha sufrido un cambio muy importante.
YY: Indica si la versión es de desarrollo o de producción. Un número impar, significa que es de desarrollo, uno par, que es de producción.
ZZ: Indica nuevas revisiones dentro de una versión, en las que lo único que se ha modificado, son fallos de programación/bugs.

La tercer capa esta conformada por el grupo de los Shells o interpretes de comandos, los cuales funcionan como la interfaz entre el usuario y el Kernel, proporcionan las herramientas para que el usuario se pueda comunicar con el núcleo del sistema de Linux.

La cuarta y última capa es donde se encuentra el usuario junto con los programas y aplicaciones que se le han agregado al sistema como hojas de cálculo, lenguajes de programación, manejadores de bases de datos, procesadores de texto, etc.

Sistema de archivos

Linux está compuesto de un sistema de archivos jerárquico (tabla 2.2) en el cual no existen unidades de disco, en su lugar cada unidad de almacenamiento así como cada dispositivo de hardware son reconocidos como un archivo o directorio dentro del sistema.

/		Raíz del sistema
↳	/var	Variables del sistema
↳	/var/spool/mail	Archivos de correo electrónico
↳	/var/log	Bitácoras del sistema
↳	/dev	Dispositivos de hardware
↳	/etc	Archivos de configuración del sistema
↳	/home	Directorios de los usuarios
↳	/bin	Comandos del sistema
↳	/sbin	Comandos del sistema
↳	/boot	Archivos de inicio del sistema
↳	/mnt	Dispositivos montados

Tabla 2.2. Sistema de archivos Linux.

Elementos básicos de una cuenta de usuario

Una cuenta de usuario está basada en un login y un password. El login es el identificador del usuario dentro del servidor, dicho login es único. El password es la clave de acceso al sistema, sólo debe conocerla el dueño de la cuenta ya que permite el acceso al servidor, a todos sus servicios disponibles y a la información del usuario.

passwd Comando que nos permite realizar el cambio de contraseña de una cuenta.

Salida del sistema
Es necesario indicarle al sistema que queremos terminar la sesión de trabajo en el servidor, para ello existen varios comandos, entre ellos se encuentran los comandos exit y logout o la combinación de teclas Ctrl+d.

Comandos básicos
Su estructura básica es la siguiente: comando -opciones argumento1 argumento2...

La estructura de los comandos es muy rígida y generalmente siguen la misma sintaxis, es decir, siempre se escribe primero el comando, en seguida las opciones si se requieren y al final los argumentos necesarios.

Las opciones modifican el funcionamiento de un comando, generalmente se utilizan para agregar información a su salida o aumentar el potencial del mismo, las opciones no son necesarias para la ejecución del comando, siempre son antecedidas de un guión y se puede utilizar más de una opción. Los argumentos son requisito para la ejecución del comando, si no damos el argumento apropiado, éste no se ejecutará y el sistema mandará un mensaje de error.

hostname Muestra el nombre del servidor.

Opciones

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

uname Nos indica el tipo de Linux que se esta utilizando.

-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.

pwd (print working directory) Imprime en pantalla el directorio actual de trabajo.

touch Este comando nos permite crear archivos vacíos, de tamaño 0. Es útil cuando nos interesa generar archivos que tienen una función especial como las bitácoras.

Caracteres especiales. Cada uno de ellos realiza una función en específico, en ocasiones pueden ser combinados para aumentar el potencial del comando en ejecución o para redireccionamiento, se recomienda que estos caracteres no se utilicen al nombrar archivos o directorios.

; , < > | >> ^ $ & / ~ \ [ ] ‘ “ ( ) . * ?

mkdir (make directory) Permite crear directorios, es posible crear varios directorios en la misma línea de comando escribiendo uno a continuación de otro separados por un espacio.

-p Crea todos los directorios padres inexistentes, es decir, primero busca la rama de directorios y si no la encuentra la crea.
-v Informa sobre la creación de directorios.

rmdir (remove directory) Borra directorios, la única condición que debe cumplir es que se encuentre vacío el directorio a ser borrado.

ls (list) Muestra un listado del contenido del directorio actual de trabajo.

-l Muestra un listado con formato largo indicando todas las propiedades de los archivos y directorios del directorio actual de trabajo.
-a Muestra todo el contenido del directorio, incluyendo los archivos y directorios ocultos.

-r Muestra el listado en orden inverso alfabético.
-R Muestra un listado en forma recursiva, es decir lista el contenido de directorios y subdirectorios a partir de la ruta donde le indiquemos.
-t Muestra un listado tomando como opción la fecha de modificación del archivo o directorio.

Comodines *, ? Existen dos caracteres llamados comodines los cuales pueden sustituir un carácter o conjunto de caracteres durante la ejecución de comandos, Ejemplo:

ls ??a Muestra aquellos archivos y directorios los cuales tengan el nombre de tres letras y la tercera sea la letra a.
ls *u*t* Muestra aquellos archivos y directorios los cuales tengan en el nombre una u y una t en cualquier posición pero en el orden que se le indica.

cd (change directory) Permite movernos entre directorios, es necesario que le indiquemos la ruta del directorio al que nos queremos cambiar. Es posible utilizar rutas absolutas o relativas según sea nuestra conveniencia.

Las rutas absolutas siempre inician con el símbolo de / que indica el punto más alto de la estructura de directorios, a partir de aquí se debe de poner la ruta completa para llegar al punto que se quiera afectar.

cd /home
cd /var/spool/mail

Las rutas relativas nunca inician con el símbolo de raíz / en lugar de ello inician la ruta a partir del directorio actual de trabajo. Supongamos que nos encontramos en el directorio home y queremos regresar dos directorios hacia arriba para posicionarnos en la raíz del sistema:

cd ../..

cp (copy) Nos permite hacer copias de archivos uno por uno, múltiples archivos o incluso hacer copias de estructuras de directorios.

cp ruta_origen ruta_destino

-r Copia recursiva.
-i Indica al sistema que realice una copia interactiva de tal forma que si se esta copiando un archivo con un nombre de un archivo que ya existe, el sistema pedirá confirmación para sobrescribir el archivo existente.
-v Nos va informando del nombre de los archivos a los cuales se les esta haciendo la copia.

rm (remove) Borra archivos, puede ser borrado de un solo archivo o varios haciendo uso de los metacaracteres (* ?).

-r Borrado en forma recursiva, esta opción es de mucho cuidado ya que se borrará el directorio especificado y todo su contenido.
-f Fuerza a que los elementos sean borrados.

mv (move) Tiene dos funciones básicas, la primera es mover archivos o estructuras de directorios de un lugar a otro, la segunda es renombrar archivos o directorios.

man Manual del sistema en línea de los comandos, su argumento es el comando que queremos consultar. Ejemplo: man ls.

También es posible acceder al manual del manual para obtener ayuda respecto al funcionamiento del comando man. Ejemplo: man man.

Una forma rápida de obtener ayuda de un comando es tecleando el comando seguido de doble guión más la letra h (de help): ls –h.

Esto desplegará en pantalla únicamente las opciones disponibles del comando ls sin mostrar la ayuda completa que, se compone de 8 secciones:

- Herramientas de usuario.
- Llamadas al sistema.
- Llamadas a bibliotecas C.
- Información de controladores de dispositivos.
- Archivos de configuración.
- Juegos.
- Paquetes.
- Herramientas de sistema.

Otros comandos que muestran información relacionada con los comandos del sistema son:

apropos Muestra las secciones del manual que contiene instancias de una palabra clave.
whatis Describe, hace búsquedas del comando y da el encabezado de la sección del manual.

Trabajo con archivos

cat nos permite ver el contenido de uno o más archivos, su función original es concatenar archivos.

cat archivo1 archivo2 ...

-n Imprime el contenido del o los archivos numerando las líneas en forma ascendente.

more Este comando nos permite ver el contenido de uno o más archivos imprimiéndolos en pantalla, para avanzar una pantalla se presiona la barra espaciadora, para avanzar una línea se presiona la tecla de enter.

less La orden more es bastante hábil, pero a menudo encontraremos que nos hemos adelantado más allá de la pantalla que queríamos. more no proporciona una manera de retroceder. El comando less proporciona esa función.

head Nos muestra las primeras 10 líneas contenidas en un archivo

head nom_archivo

-n Despliega las primeras n número de líneas.

tail Nos muestra las últimas 10 líneas contenidas en un archivo

tail nom_archivo

-n Despliega las últimas n número de líneas contenidas en un archivo.
+n Despliega todas las líneas contenidas en un archivo, a partir de la línea número n.

pico Este es un programa que permite editar archivos de texto, es muy fácil de utilizar, ya que presenta un menú de comandos en la parte inferior de la pantalla y no requiere de muchos comandos para poder editar el contenido del archivo.

Redireccionamiento. En Linux existe una salida estándar que es el monitor, de igual manera hay una entrada estándar que es el teclado, de tal modo que todo lo que queramos hacer en forma convencional se lo debemos indicar al sistema operativo.

Redireccionamiento de salida. El carácter mayor que (**>**) nos permite redireccionar la salida de un programa hacia un archivo, ejemplo: ls > milista.

Manda la salida del comando ls a un archivo llamado milista, el resultado del comando ls no se imprime en pantalla, sino en el archivo indicado.

Redireccionamiento de entrada. El carácter menor que (**<**) nos permite redireccionar un archivo hacia un programa, ejemplo: mail usuario < milista.
Manda el contenido del archivo milista por correo al usuario con login usuario.

Redireccionamiento de un programa a otro programa (Tuberias). El carácter pipe (|) funciona como un filtro mandando la salida de un programa hacia la entrada de otro, ejemplo: ls –l | more.
Aquí se está enlazando la salida del comando ls con el programa more, el cual nos mostrará el resultado del comando ls por pantalla cada vez que presionemos la tecla espaciadora.

Redireccionamiento no destructivo o de adición. Doble mayor que (**>>**) nos permite 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 añade al final sin eliminar el contenido del archivo, ejemplo: ls >> milista.

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>error
Aquí, la salida de verdad esta direccionada al archivo milistado mientras que la salida de error esta direccionada al archivo error.

Filtros

grep Busca patrones en archivos. Por defecto devuelve todas las líneas que contienen un patrón determinado en uno o varios archivos. Sintaxis: grep [opciones] <patrón> [ficheros].

-c Devuelve sólo la cantidad de líneas que contienen al patrón.
-i Ignora las diferencias entre mayúsculas y minúsculas.
-H Imprime además de las líneas, el nombre del archivo donde se encontró el patrón. Es así por defecto cuando se hace la búsqueda en más de un archivo.
-l Cuando son múltiples archivos sólo muestra los nombres de aquellos donde se encontró al patrón y no las líneas correspondientes.
-v Devuelve las líneas que no contienen el patrón.
-r Busca en un directorio de forma recursiva.
-n Imprime el número de cada línea que contiene al patrón.

Tipos de archivos

Dentro del sistema operativo vamos a encontrar archivos de diferentes tipos, los cuales los vamos a distinguir por el primer bit de sus propiedades. El atributo del primer bit nos indica si el elemento listado es directorio o archivo, así mismo nos indica de qué tipo de archivo se trata.

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

Permisos. Existen tres tipos de permisos que se aplican tanto a los archivos como a los directorios, ellos son: lectura (r), escritura (w) y ejecución (x).

r El permiso de lectura permite que el archivo pueda ser leído o copiado.
w Permite escribir en el archivo, hacer modificaciones o borrarlo.
x Los archivos ejecutables son aquellos que pueden realizar un proceso.

Estos permisos se encuentran asignados en tres 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, de ello que los atributos se componen de 10 bits, el primer bit indica el tipo de elemento que esta listado (archivo o directorio), los siguientes 9 bits representan los tipos de permisos asignados a cada elemento listado.

chmod Nos permite modificar los permisos a los archivos y directorios, la asignación de permisos puede hacerse de dos formas, mediante el método simbólico o mediante el método octal.

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 otros usuarios que no pertenecen al grupo.
a Indica que afectará los permisos para todos los usuarios.

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

Para modificar los permisos se pone el indicador de los permisos a afectar ya sea para el usuario, el grupo o los otros, seguido del signo + si se quiere agregar el permiso o de – si se quiere quitar el permiso separando con una coma los bloques de permisos, al final se indica el nombre del elemento a afectar.

Método octal

X	vale	1
W	vale	2
R	vale	4

Quedando nuestra tabla de la siguiente forma:

Asignación		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

De tal forma que si deseamos poner todos los permisos para el usuario, permisos de lectura y ejecución para el grupo y ninguno para los otros, el comando quedaría de la siguiente forma:

chmod 750 hola

Comunicación con los usuarios

who Muestra un listado sencillo de quien se encuentra conectado en el servidor, a qué hora se conecto y qué está haciendo.

w Muestra un listado con detalles de los usuarios conectados en el servidor.

whoami Indica con qué login estamos registrados dentro del servidor, este comando es útil ya que es posible que un usuario se convierta en otro diferente utilizando el comando su.

finger Muestra información de los usuarios conectados en el servidor, entre ella el nombre real del usuario.

Correo electrónico

mail cuenta@mail.com. El comando mail nos permite enviar correo electrónico a cualquier usuario que tenga acceso a una cuenta de e-mail.

Opciones de revisión de mail

h Nos mostrará una lista numerada de los títulos de los correos recibidos.
Tecleando el número que aparece en el listado de correos veremos el contenido del correo deseado.
r Manda una replica del correo que estamos visualizando en ese momento.
q Sale del programa mail guardando todos los cambios que hayamos realizado como borrado de correos, marcado de correos ya leídos, etc.
x Sale del programa mail sin salvar cambios en la lista, es decir, que no marcará los correos que ya consultamos y los mantendrá marcados como nuevos.
d# Borra el correo marcado en la lista con el número que le estamos indicando.

pine Al igual que mail, nos permite trabajar con el servicio de correo electrónico pero este programa tiene muchas ventajas en comparación con el comando mail, presenta un menú de opciones, configuración del servició de correo, envío de archivos adjuntos, etc.

write Permite enviar un mensaje a un usuario, siempre y cuando el usuario se encuentre conectado en el servidor.

talk Este programa permite realizar una conversación interactiva en tiempo real.

wall Nos permite enviar un mensaje a todos los usuarios que se encuentran conectados al servidor, el mensaje es escrito directamente en pantalla.

scp Permite copiar información de una máquina remota a otra.

Algunos programas útiles dentro de Linux

fortune Elige de una base de datos el mensaje del día como el que aparece en pantalla cada vez que iniciamos sesión.

date Muestra la hora del sistema.

df Muestra el estado de las particiones de los discos duros indicándonos el espacio utilizado y el estado libre en disco.

du Muestra el espacio utilizado entre archivos y directorios a partir del directorio desde donde se ejecutó el comando.

last Muestra las últimas conexiones de los usuarios al servidor.

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

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

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

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

banner Este programa permite hacer carteles de tipo banner, por default tiene un ancho de 132 líneas lo que impide que el banner pueda ser leído directamente en pantalla; para ello se debe utilizar la opción –w# para cambiar el número de columnas que abarcará el mensaje.

banner –w35 saludos a todos.

cal El sistema cuenta con un calendario el cual comprende desde el año 1 hasta el año 9999, por default el comando cal nos va a entregar el calendario del mes en curso.

find Busca un archivo a partir de un directorio.

Procesos. Un proceso es un programa que se encuentra corriendo dentro del servidor, cada proceso tiene un tiempo de vida que va desde el momento en que presionamos la tecla de enter hasta que termina la ejecución del proceso.

ps Permite ver los procesos que se están ejecutando desde nuestra sesión abierta, y que se encuentra en uso en este momento, muestra un listado de los procesos en formato corto.

ps –u login Muestra un listado de todos los procesos que está ejecutando un usuario, aunque se encuentre trabajando en varias sesiones, el sistema listará todos los procesos que le pertenecen a dicho usuario.

ps –f Entrega un listado de los procesos que se están ejecutando en nuestra sesión, muestra los procesos en formato largo, donde podemos ver la dependencia de los procesos por el UID, PID y el PPID.

- El UID es el identificador del Usuario dueño del proceso.
- El PID es el número del identificador del Proceso en ejecución.
- El PPID es el identificador del Proceso Padre del Proceso en ejecución.

ps –e Muestra un listado en formato sencillo de todos los procesos ejecutándose en el servidor.

ps –fea Muestra todos los procesos ejecutándose en el servidor en formato largo.

ps –h nos indica el estado en que se están ejecutando los procesos.

Estados de los procesos:

Z Zombie, cuando un proceso corre en forma independiente, incluso del shell o del proceso que lo invocó.
R En ejecución, se encuentra en la CPU.
T Stopped o detenido.
S Sleeping, durmiendo en espera de recibir una solicitud para iniciar su trabajo.
D Dormido sin interrupción posible, normalmente relacionado con entrada/salida.

Procesos en primer y segundo plano

jobs Nos permite ver los programas que se están ejecutando en segundo plano, nótese que a cada proceso en segundo plano se asigna un número que nos sirve para identificar el proceso.

-l Muestra los procesos en segundo plano con formato largo.

Es posible enviar un proceso a segundo plano en forma automática desde el momento en que éste se inicia, esto se logra colocando el carácter de ampersan (&) al final de la línea del comando.

find / -name shadow 1> usuarios &

Para mandar un proceso a primer plano debemos dar la instrucción fg # donde el # representa el número que nos entrega el comando jobs.

kill Nos permite entre otras opciones matar procesos, nosotros como usuarios únicamente podemos matar nuestros propios procesos.

Respaldos

tar Nos permite empacar una estructura de directorios, su función principal es almacenar el contenido de un directorio en forma recursiva dentro de un archivo.

Sintaxis: tar [opciones] archivo.tar archivo

-c Crea un nuevo archivo tarareado.
-v Verbose, imprime en pantalla todo aquello que está afectando.
-f Indica al sistema que va a afectar a un archivo
-x Extrae el contenido de un archivo .tar
-t Lista todos los archivos contenidos en un archivo .tar en formato largo mostrando todos sus detalles.

Como regla general cualquier opción es combinada con las opciones –vf ya que es recomendable ver qué está realizando el sistema, así como para indicarle que se está enviando la salida del comando a un archivo, de lo contrario tomará por default la unidad de cinta.

gzip Permite realizar una compresión de archivos, manejando 9 niveles de compresión que van desde el 1 al 9, donde el nivel 1 es el nivel de compresión más bajo mientras el nivel 9 es el nivel de compresión más alto. Añade la extensión .gz al archivo comprimido.

Sintaxis: gzip [-n] archivo

-n Indica el nivel de compresión que se quiera realizar.
-d Esta opción permite descomprimir un archivo gzipeado. Sintaxis: gzip -d archivo.
-f Fuerza la compresión o descompresión de la información afectada sin importar los permisos o propiedades que tenga el archivo o directorio al momento de ser afectado.
-t Hace una revisión del archivo tarareado para verificar su integridad.
-l Muestra las propiedades del archivo tarareado indicando el tamaño del archivo cuando esta comprimido, el tamaño de la información cuando se encuentra descomprimida y el porcentaje de compresión de la información. gzip –lv archivo.tar.gz.

Ligas. Existen dos tipos de ligas, una de ellas es la de tipo suave y la otra es de tipo dura. La liga de tipo suave es un vínculo hacia un archivo pero la liga es de tamaño muy pequeño ya que sólo hace referencia al archivo que se esta 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.

ln –s nom_archivo nom_liga

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

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, la liga está asociada al mismo inodo[3] del archivo con el que está ligado.

ln nom_archivo nom_liga

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

2.2 Módulo II. Instalación y administración de Linux

En este módulo hablaremos acerca de las recomendaciones al instalar Linux y la importancia de una buena administración del sistema.

El administrador

El administrador de sistemas es la persona responsable de **configurar**, **mantener** y **actualizar** el sistema. Cuidando el funcionamiento del software, hardware y periféricos de forma que estén disponibles para ser utilizados por los usuarios.

Sus actividades

- Proporcionar un ambiente seguro, eficiente y confiable.
- Brindar un funcionamiento confiable del sistema.
- Planear las actividades.
- Tener copias de seguridad (información de los usuarios, bases de datos, correo electrónico y archivos originales del sistema).

Para ello es indispensable que cumpla con un perfil adecuado, entre otras cosas, debe conocer las utilerías básicas del sistema, como son, cut, sort, paste, diff, comm, tail, head, grep, egrep, compress, etc. Control de tareas (at, crontab), respaldos (dump, dd, restore, tar, cpio). Manejar herramientas de programación como C, shell, awk o Perl será fundamental.

En general, debe conocer al menos un lenguaje de programación, funcionamiento del sistema operativo, técnicas de administración, conocimientos básicos de hardware y mantenimiento de dispositivos, comprensión profunda sobre redirección, tuberías, procesamiento en segundo plano, manejo de vi (común denominador en cuanto a editores en Linux), programación shell, etc. (En el anexo 1 se muestra una lista de los comandos básicos del editor vi).

Políticas

Las políticas son guías para orientar la acción; son criterios, lineamientos generales a observar en la toma de decisiones, sobre problemas que se repiten una y otra vez dentro de una organización. En este sentido, las políticas son criterios generales de ejecución que auxilian el logro de objetivos y facilitan la implementación de las estrategias, habiendo sido establecidas en función de éstas.

[3] Estructura de datos propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo Unix como es el caso de Linux. Un inodo contiene el tamaño de un archivo, las tres fechas relacionadas con el archivo (modificación del contenido, acceso al contenido y modificación del inodo), los permisos, el dueño, el grupo, el número de links, e información sobre dónde están físicamente los datos (los bytes) del archivo en el disco.

Lineamientos para su formulación:

- Establecerse por escrito, y dársele validez.
- Redactarse claramente y con precisión.
- Darse a conocer a todos los niveles donde se va a interpretar y aplicar.
- Coordinarse con las demás políticas.
- Revisarse periódicamente.
- Ser razonable y aplicable a la práctica.
- Estar acorde con los objetivos.
- Debe ser estable en su formulación.
- Ser flexible.

Ejemplo de política: No se permite cualquier operación no expresamente autorizada. Sólo se permite el acceso de acuerdo al rol definido por el administrador el cual corresponde a su perfil. Como ejemplo, el usuario de un programa únicamente necesita disponer de permisos para leer y escribir los archivos utilizados por esa aplicación concreta, pero nada más.

Procedimientos

Los *procedimientos* describen, en forma de lista o receta, la forma en que se realizan las diferentes tareas. Son útiles tanto para los administradores nuevos como para los experimentados. Entre sus múltiples ventajas, se destacan:

- Las tareas se realizan siempre del mismo modo.
- Se reducen las probabilidades de error.
- Es más rápido trabajar siguiendo una receta.
- Los cambios quedan documentados en el propio procedimiento.

Deben existir procedimientos para las siguientes tareas:

- Agregar, eliminar, bloquear una cuenta de usuario.
- Agregar una máquina.
- Localizar una máquina.
- Instalar "TCP wrappers" para registro y control de accesos vía TCP (telnet, ftp, finger).
- Instalar respaldos para una máquina nueva.
- Configurar la seguridad de una máquina nueva.
- Prever el rearranque de piezas complejas de software.
- Revivir un sitio Web que no responde o no sirve las páginas.
- Destrabar y rearrancar una impresora.
- Actualizar el sistema operativo.
- Instalar un paquete de software.
- Instalar software desde la red.
- Actualizar paquetes críticos de software (sendmail, gcc, named, etc.).
- Respaldar y restaurar archivos.
- Definir condiciones y alcance en bajadas de emergencia, etc.

El administrador de sistemas es una persona con poder: dispone de la contraseña de supervisor, puede leer el correo de los usuarios, borrar o alterar sus archivos, distorsionar intencional o accidentalmente el sistema hasta dejarlo inutilizable, producir pérdida de datos, etc. El trabajo y respeto por los individuos y la institución es fundamental. La persona elegida para administrar un sistema debe tener firmes convicciones éticas. Alguien con tendencia a presumir de su condición privilegiada debe ser invitado prontamente a cambiar de ocupación.

Las condiciones de un buen administrador son un tanto contradictorias: debe ser innovador en la búsqueda de soluciones, pero cuidar de no arriesgar el sistema; debe ayudar a los usuarios pero sin resentir la atención comunitaria; debe ser amable, pero firme en los momentos críticos.

Instalación de Linux

Para poder elegir una distribución de Linux utilizamos varios criterios.

En cuanto al hardware del equipo: La arquitectura de la máquina, el tipo de procesador, capacidad del disco duro, memoria RAM, etc.

En cuanto al uso: Si se piensa adquirir licencia, las versiones estables de la distribución y sus características, uso que se le dará al equipo (uso personal, servidor de correo, servidor de aplicaciones, servidor Web, servidor de bases de datos, servidor de impresión, servidor de archivos NTFS, etc.), que sea compatible con las aplicaciones a instalar, o no presente demasiados bugs, etc.

En el caso de Slackware Linux no se requiere de un sistema extremadamente potente. Se puede ejecutar en equipos 386 o superiores. Los requerimientos mínimos ya se mencionaron en el módulo anterior.

El conjunto de paquetes con el que cuenta Slackware se reparten en series ordenadas alfabéticamente:

A	Es la base, contiene el Kernel y utilidades básicas del sistema.
AP	Varias aplicaciones que no requieren del sistema X Windows.
D	Herramientas de desarrollo de programas, compiladores, depuradores, intérpretes como C, Perl, Lisp, etc.
E	El editor GNU Emacs, es tan grande que requiere su propia serie.
F	Contiene FAQs, HOWTOs, y documentación esencial para la administración.
GNOME	Contiene el ambiente de escritorio de GNOME y librerías relacionadas.
K	El código fuente para el núcleo de Linux.
KDE	Contiene el ambiente de trabajo de escritorio KDE y QT.
KDEI	Contiene el conjunto de idiomas internacionales para KDE.
L	Librerías del sistema necesarias para los escritorios GNOME, KDE y más.
N	Contiene programas relacionados con la red. Demonios, clientes, etc.
T	Software de tipo TeX usado generalmente para documentos técnicos.
TCL	Contiene las herramientas del lenguaje de comandos, el Tk, el TclX, y el TkDesk.
X	Contiene la base del sistema X Window.
XAP	Contiene una colección de aplicaciones para X Por ejemplo Ghostscript y Netscape.
Y	Contiene juegos (una colección de juegos de BSD).

Si nuestro equipo cuenta con la opción de arranque por medio de un CD-ROM, esta sería la forma más fácil de instalar Linux Slackware. Aparecerá una pantalla de bienvenida al programa de instalación de Linux, en la parte inferior de esta pantalla aparecerá el indicador boot como lo muestra la figura 2.2:

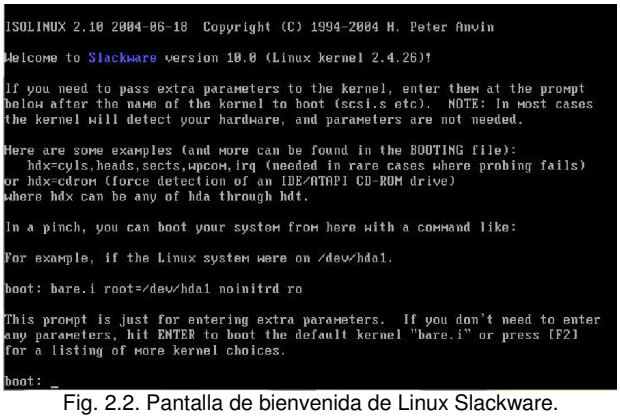


Fig. 2.2. Pantalla de bienvenida de Linux Slackware.

El primer paso de la instalación es elegir la configuración de nuestro teclado. Debemos elegir la opción es.map del menú, que es el mapa del teclado para la configuración en español. El termino mapa del teclado se refiere a la ubicación de las teclas y dependiendo del idioma éstas tendrán una ubicación diferente (figura 2.3). Elegimos la opción qwerty/es.map para el mapa en español.

Fig. 2.3. Eligiendo el mapa del teclado.

Debemos probar la localización de los caracteres en el teclado para verificar que la distribución corresponde con el tipo de mapa que elegimos. A continuación, tenemos que registrarnos (como root) ante el sistema para que nos entregue un shell y seguir con la instalación.

Lo siguiente es crear las particiones necesarias para la instalación. El administrador debe tener un esquema de disco pensado para una administración óptima y que sea flexible para facilitar cualquier tarea que se desee realizar, pensando en la funcionalidad del servidor. Tener todo el sistema en una sola partición es mala idea ya que no es flexible y hace difícil de administrar el servidor.

Lo más recomendable es hacer particiones pensando en la carga de archivos del sistema, de los usuarios, servicios y otras consideraciones, logrando con ello facilitar las tareas de respaldo, actualización, administración fácil y eficiente, entre muchas otras ventajas.

Para particionar el disco usamos el comando: **fdisk** /dev/hda.

Si utilizamos el comando indicándole el disco apropiado nos mostrará una pantalla como la de la figura 2.4.

Fig. 2.4. Particionando el disco.

Una vez realizadas las particiones necesarias, tenemos que guardar los cambios y comenzar la instalación con el comando **setup** (figura 2.5).

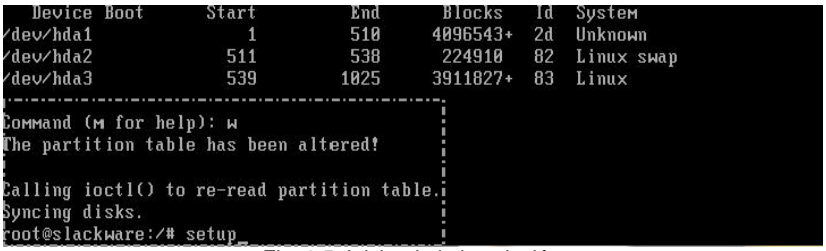


Fig. 2.5. Iniciando la instalación.

Se despliega el asistente de instalación ahora en formato gráfico en forma de menú. Tenemos que seguir el procedimiento de arriba hacia abajo, lo primero es elegir el tipo de teclado como anteriormente se realizó.

Una vez que se ha definido el tipo de teclado el sistema automáticamente detecta la partición de Swap, nos solicita darle formato y la prepara para la instalación. Una vez hecho lo anterior, el proceso de instalación reconoce las particiones de Linux, la partición que elijamos en este momento es donde se va a instalar el sistema.

Ya que elegimos la partición solicita darle formato rápido, formato con revisión de sectores dañados o montar la partición sin formatear. Si elegimos la opción de dar formato nos solicita el tipo de Sistema de Archivos con el cual va a ser formateada la partición.

Ya terminado el formato de todas las particiones entrega un reporte con información de la(s) particiones (figura 2.6).

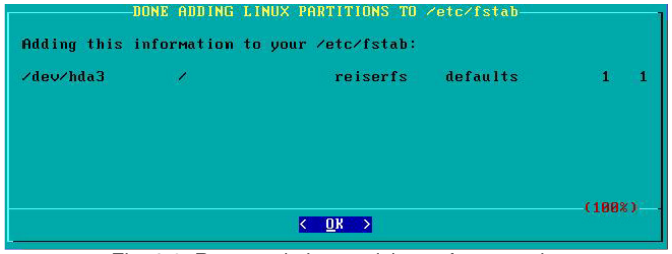


Fig. 2.6. Reporte de las particiones formateadas.

El siguiente paso es elegir el recurso desde el cual se va a realizar la instalación. En este caso le indicamos que vamos a instalar desde CD, el asistente nos mostrará un mensaje solicitando auto montaje o montaje del CD en forma manual. Lo más conveniente es decirle que aplique un montaje del CD en forma automática.

Ya que se ha detectado el CD, el asistente nos pedirá elegir las series de paquetes que se van a instalar y acto seguido el modo de instalación, el modo Full es automático, los otros modos solicitan la instalación de los paquetes ya sea por series o aplicando algún otro criterio.

Después de haber terminado la instalación de paquetes continuamos con la configuración del sistema, el siguiente paso es indicarle desde qué medio deseamos instalar el Kernel para el sistema, obviamente en este caso es desde CD.

Ahora tenemos que elegir el Kernel que vamos a instalar, éste depende de la arquitectura y de las características de la computadora donde se está instalando.

Fig. 2.7. Eligiendo el Kernel.

En seguida nos ofrece la posibilidad de hacer un disco de arranque para poder iniciar desde él el sistema, este disco puede servir como de recuperación o de emergencia para recuperar el sistema. El asistente también nos permite configurar un modem, si es que contamos con uno externo o que se encuentre en una ranura de expansión podremos configurarlo fácilmente, si es un modem integrado a la tarjeta principal generalmente no es detectado ya que pertenece a la familia de los winmodem.

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

LILO es el gestor de arranque, nos ayuda a generar un menú de inicio para equipos multisistemas, entre otras cosas, si la instalación está contemplada para una máquina que tiene dos sistemas operativos (en el caso de servidores es recomendable sólo uno) se realizará la configuración. Este menú nos permite iniciar la configuración de LILO con las propiedades principales, para ello tenemos que iniciar con la opción Begin (figura 2.8):

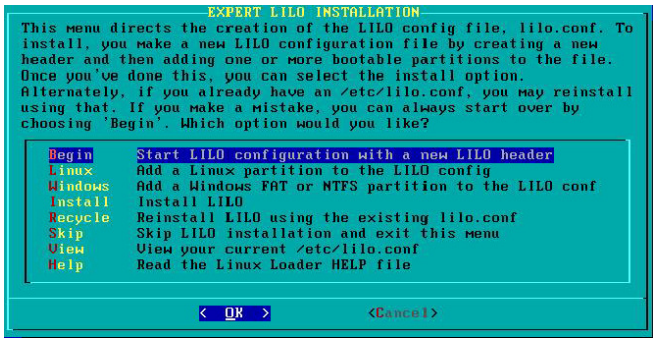


Fig. 2.8. Iniciando la configuración de LILO.

Podemos ingresar opciones extra si es que las requerimos, la mayoría de los sistemas no requieren opciones extra. Enseguida podemos elegir del menú el tipo de resolución que deseamos para modo texto o consola, para ello necesitamos saber las características de la tarjeta de video y qué tipos de resolución soporta.

El siguiente paso es indicarle en dónde deseamos instalar LILO, lo recomendable es que se instale en el MBR[4] del disco duro. También tenemos que elegir un tiempo de espera antes que inicie con el primer sistema que tiene por default.

Hay que agregar los sistemas que deseamos sean visibles en el menú de booteo de los sistemas operativos. Primero se agrega el sistema Windows (si contamos con dicho sistema).

Después tenemos que agregar Linux usando la opción de "Linux" del menú de configuración (figura 2.8). Nos muestra las particiones de tipo Linux y elegimos la partición apropiada. Tenemos que insertar la etiqueta que es el nombre que llevará en el menú de booteo (figura 2.9).

Fig. 2.9. Etiqueta de la partición Linux.

Cuando hayamos terminado de agregar los sistemas a nuestro gestor de arranque, el siguiente paso es indicarle al sistema que instale LILO.

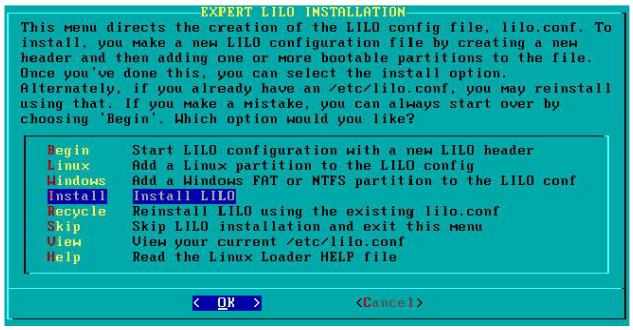


Fig. 2.10. Instalación de LILO.

Ahora tenemos que indicarle qué tipo de mouse está conectado a la computadora y habilitarlo en modo consola, es de gran ayuda para copiar y pegar en forma rápida.

Procedemos a configurar la red si es que nuestro equipo cuenta con una tarjeta soportada por esta distribución de Linux y es detectada automáticamente (figura 2.11).

Fig. 2.11. Configuración de red.

- El primer paso es poner el nombre de la máquina o HOSTNAME.
- A continuación debemos indicar el dominio de nuestra red, si es que se cuenta con él.
- Ahora debemos indicar el tipo de dirección ip que se va a configurar, para ello contamos con tres opciones, en nuestro caso elegimos static ip.
- El siguiente cuadro nos pide la dirección ip para nuestro servidor.
- Ahora debemos indicarle la máscara de nuestra red.
- El siguiente paso es indicarle nuestra puerta de enlace.

[4] Un master boot record (MBR) es el primer sector ("sector cero") de un dispositivo de almacenamiento de datos, como un disco duro.

- Ahora el asistente nos pregunta si queremos configurar un servidor de nombre de dominio o DNS.
- Finalmente nos entregará un reporte con todos los datos que hemos configurado para la tarjeta de red (figura 2.12).

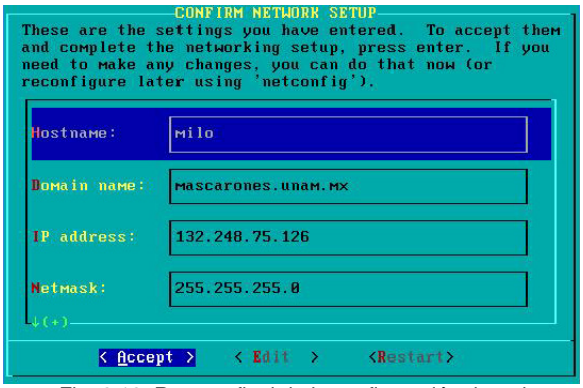


Fig. 2.12. Reporte final de la configuración de red.

El asistente nos ofrece la posibilidad de indicarle los servicios que deseamos estén en funcionamiento, si no le indicamos al sistema que se inicien desde este menú lo podemos hacer posteriormente.

Si contamos con un reloj atómico que sincronice todas las computadoras, lo podemos configurar en este momento. Es importante definir la zona horaria ya que nos ayuda a los cambios automáticos de horario por temporadas. La interfaz gráfica que elijamos en este momento es con la que va a iniciar la X por default (figura 2.13).

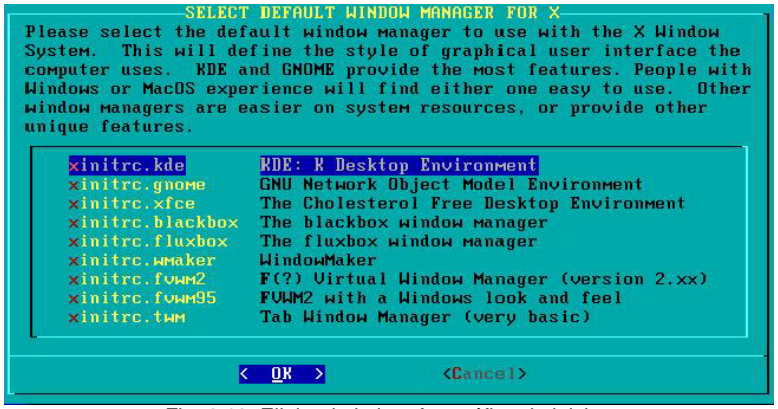


Fig. 2.13. Eligiendo la interfaz gráfica de inicio.

Es momento de teclear una buena contraseña para el usuario administrador root.

Se ha terminado la instalación y el sistema pide reiniciar el equipo presionando la combinación de teclas ctrl-alt-sup, si presionamos Enter nos regresa al menú de instalación, donde podemos elegir la opción de Exit. Eligiendo ésta el sistema extrae la charola de CD para retirar el disco de instalación, ahora podemos retirar el CD y reiniciar el equipo con la combinación de teclas ctrl-alt-sup.

Después de reiniciar el sistema debemos tener la pantalla del gestor de arranque LILO en la cual podemos elegir el sistema operativo con el cual deseamos trabajar (figura 2.14).

Fig. 2.14. Linux Slackware está instalado.

Dar de alta y/o baja el sistema

Para apagar el sistema podemos usar varios comandos:

poweroff Este comando hace un apagado inmediato del sistema, muy peligroso para ciertas tareas ya que no verifica que los procesos se hayan terminado en forma apropiada, da de baja todos los servicios y apaga el sistema, ahora este programa se ha actualizado para no causar daños al sistema o información por un apagado deficiente.

halt Muy parecido a poweroff, solo que éste para apagar el sistema hace un llamado al comando halt que en realidad es quien apaga el sistema, la desventaja de usar halt es que funciona muy parecido o poweroff.

reboot Este comando sirve para reiniciar el sistema, igualmente no realiza un apagado eficiente ya que no verifica que los programas sean terminados apropiadamente.

shutdown Este es considerado el comando más eficiente para detener y reiniciar el sistema, ya que su proceso de apagado es muy eficiente, de hecho “da de baja el sistema en una forma segura”. Se tiene control del apagado del sistema ya que se puede definir un tiempo, todos los usuarios son avisados de que el sistema se va a dar de baja mediante la instrucción SIGTERM, deshabilita el login para que nadie se pueda logear, se comunica con el programa init y le solicita el nivel 0 de ejecución.

Este comando envía una señal de sincronización de discos, manda la señal de termino a todos los programas y los cierra eficientemente siguiendo su proceso normal de las aplicaciones, una vez que se han cerrado todos los procesos manda una nueva señal de sincronización para verificar que no hay tareas en ejecución, termina, baja los demonios y finalmente apaga el sistema en forma segura.

Mantenimiento de claves de usuarios

Una de las principales tareas es la administración de las cuentas de usuario, para ello se debe tener un plan administrativo de claves contemplando altas, bajas y cambios de las claves, también es importante definir los grupos a los que va a pertenecer uno o más usuarios. Se debe tener un especial cuidado en la capacitación de los usuarios para elegir contraseñas fuertes, asignar permisos adecuados y administrar apropiadamente los recursos para no tener un abuso de ellos. Comandos para administrar usuarios:

adduser Agrega usuarios en modo comando.
useradd Agrega usuarios en modo comando.

userdel Elimina usuario.
usermod Modifica las propiedades del usuario.
groupadd Agrega grupos.
groupdel Elimina grupos.
groupmod Modifica grupos.
groups Lista los grupos a los que pertenece un usuario.
kuser Administra usuarios y grupos en modo gráfico.

Uso de recursos:
du (disk usage) Permite monitorear el estado del sistema de discos.
df Muestra el espacio ocupado por directorios entregando un total de espacio ocupado, este comando se puede aplicar sobre los directorios de cada usuario y saber rápidamente cuánto espacio está usando cada usuario en total.

Instalación y mantenimiento de dispositivos

El sistema dispone de un directorio /mnt el cual forma parte de la estructura estándar de Linux, dentro de este directorio existen algunos otros que se sugieren para montar las unidades de disco. Cuando instalamos Linux, éste reconoce todas las unidades de disco y crea un directorio para cada una de ellas en /mnt, si agregamos más unidades de disco al CPU nosotros podremos crear tantos directorios como necesitemos.

ls /mnt
cdrom
floppy
hdb
hdc

El sistema cuenta con apuntadores que nos ayudan a ciertas tareas, tal es el caso de la unidad de CD-ROM, en /dev/ existe un apuntador que se dirige al directorio /mnt/cdrom, el cual está declarado dentro de un archivo de configuración, en este caso si deseamos montar el CD de forma rápida podemos hacer: mount /dev/cdrom. Cabe aclarar que son pocos los dispositivos que cuentan con estos apuntadores, nosotros podemos generarlos para agilizar sus tareas.

Para desmontar un dispositivo debemos seguir ciertos lineamientos, primero el dispositivo no debe estar en uso, esto quiere decir que nadie debe estar realizando tareas como lectura o escritura, otra es que una unidad no se puede desmontar si alguien se encuentra colocado en la estructura de directorios de dicha unidad.

umount /mnt/hdc o umount /dev/hdc

Uso de memoria y CPU

Son pocas las utilerías que permiten controlar el uso de dispositivos como la memoria y el CPU, hay que aplicar parches del sistema e incluso recompilar el Kernel.

free nos permite monitorear el estado de la memoria tanto física como la partición de Swap (figura 2.15).

```
carlos@casa:~$ free
             total       used       free     shared    buffers     cached
Mem:        239916     236052       3864          0      65536      83828
-/+ buffers/cache:      86688     153228
Swap:       265032          0     265032
carlos@casa:~$
```

Fig. 2.15. El comando free.

Es necesario saber qué programas son los que están consumiendo mayor cantidad de procesador, para ello podemos obtener un monitoreo en tiempo real del procesador y procesos en ejecución con el comando **top**.

Interfaces gráficas

Linux incorpora varias interfaces gráficas, entre las cuales podemos elegir la que queremos utilizar por medio de un menú desplegable al momento de iniciar sesión en la interfaz gráfica, cada una de ellas tiene sus propias cualidades, sin embargo, hay dos que tienen mayor aceptación por los usuarios, ellas son KDE y GNOME, destacando KDE por ser la interfaz estándar sobre Linux, es muy flexible y de fácil manejo.

Cuando iniciamos la interfaz KDE por primera vez aparece un asistente de configuración, en el cual podemos definir el país e idioma, entre otras opciones (figura 2.16).

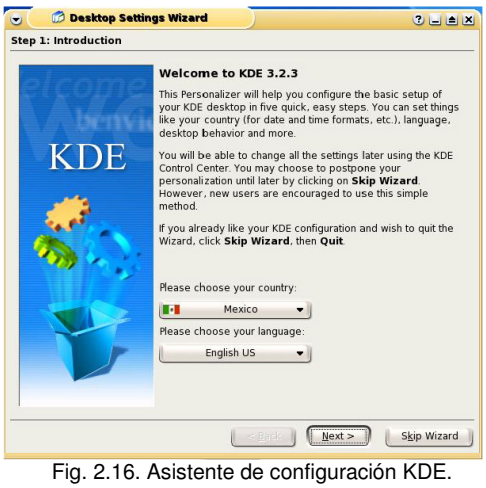


Fig. 2.16. Asistente de configuración KDE.

Si iniciamos KDE y posteriormente instalamos el idioma, es posible volver a ejecutar el asistente mediante el comando **kpersonalizer**.

También hay que configurar el teclado y el mouse si es necesario, ya que por default se encuentra configurado el teclado en inglés y el mouse como PS/2, para ello editamos el archivo /etc/X11/xorg.conf.

Configuración y mantenimiento del uso de la red

Para configurar nuestra tarjeta de red utilizaremos un comando llamado **netconfig** el cual nos llevará por medio de un asistente.

Podemos comprobar nuestra configuración con el comando **ifconfig**.

2.3 *Módulo III. Editores para la creación de páginas Web*

Ahora que tenemos instalado nuestro sistema operativo, aprenderemos a crear páginas Web con HTML. Esta herramienta, que más adelante complementaremos con PHP, es fundamental para la creación de la capa de presentación de un sistema.

La World Wide Web, la Web o WWW, es un sistema de navegador Web para extraer elementos de información llamados "documentos" o "páginas Web". Podemos referirnos a "una Web" como una página, sitio o conjunto de sitios que proveen información por los medios descritos, o a "la Web", que es la enorme e interconectada red disponible prácticamente en todos los sitios de Internet. Ésta es parte de Internet, siendo la WWW uno de los muchos servicios ofertados en la red Internet.

Los editores de HTML son un software que facilita al usuario la creación de páginas de Web, empleando para su construcción la filosofía de los procesadores de palabra que consiste en permitir que el usuario vaya capturando el texto y después, mediante las acciones de marcar y seleccionar opciones o botones, se va dando formato y presentación, además de que cuenta con una serie de asistentes que facilitan al usuario la inserción de imágenes y vínculos (ligas de hipertexto) hacia otras páginas y servicios.

Las etiquetas de HTML son el conjunto de instrucciones o comandos que después son interpretados por los visualizadores de Web (figura 2.17), desplegando como resultado de esta interpretación las páginas Web. Las Etiquetas de HTML se interpretan en el orden en que se ponen y van encerradas entre pico paréntesis (< >). Estas etiquetas no son sensitivas de tal forma que se pueden escribir con mayúsculas o minúsculas indistintamente.

Estructura básica de una página de HTML

Las páginas de HTML pueden escribirse en cualquier editor que maneje formato de sólo texto (como **vi**), o bien apoyándose en un editor de HTML que maneje ya el formato del archivo de texto por default (como blueFish). La estructura básica de una página debe llevar el siguiente esqueleto o estructura base:

```
<html>
  <head>
    <title>Espacio para título de la ventana</title>
  </head>
  <body>
    Espacio para escribir todo el contenido de la página.....
  </body>
</html>
```

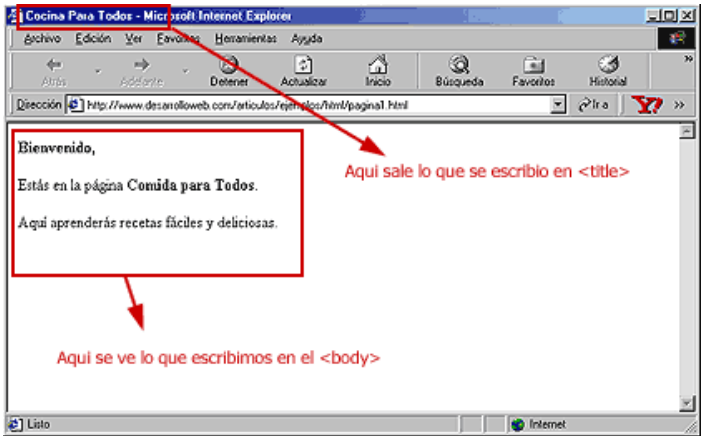


Fig. 2.17. Estructura de una página HTML.

<html>...</html> Es la primera y la última etiqueta en un documento HTML y le indica al navegador que despliegue el documento como hipertexto y no como texto sencillo.

<head>…</head> La etiqueta encabezado identifica la sección inicial del documento HTML. Entre otras cosas escribe el titulo del documento de esa sección.

<title>...</title> La etiqueta title contiene el titulo que aparece en la barra de titulo del navegador Web.

<body>...</body> La etiqueta de cuerpo abarca la parte más grande de una página Web. Contiene todo lo que el usuario ve cuando entra a esa página, la mayoría de las etiquetas se colocan dentro de body.

<hn>...</hn> ***para n=1..6*** La etiqueta encabezado crea varios tamaños de encabezados. La n se reemplaza por un número entre 1 y 6, cuanto menor sea el número mayor será el encabezado.

<hr> La etiqueta de regla horizontal, dibuja una línea horizontal en medio de la página Web, haciendo más fácil su lectura.

**
** La etiqueta de interrupción origina un salto de línea, con excepción de que coloca una línea en blanco como separador obligadamente.

<p> La etiqueta de párrafo permite separar bloques de texto y hacerlos más legibles. Cada una de estas etiquetas se coloca al principio o al final de cada párrafo.

** ... comentario ...** Esta etiqueta de vínculos permite incorporar ligas o vínculos hacia otras páginas del WWW u otros URL´s (**L**ocalizador **U**niforme de **R**ecursos).

<img src="imagen"> Esta etiqueta permite incorporar imágenes.

<blink> ... </blink> Esta etiqueta hace que el texto aparezca con un parpadeo.

<pre>... </pre> Etiqueta de texto preformateado, esta etiqueta despliega el texto exactamente como fue escrito, respetando múltiples espacios en blanco y saltos de línea, sin embargo el texto aparece con formato de letra monoespaciada.

Encabezados o headers

La etiqueta de encabezados permite resaltar un texto que se representa en la página como un titulo o encabezado. La etiqueta puede ser **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>** y **<h6>**, donde <h6> le aplica al texto el tipo de letra más pequeño y <h1> el más grande.

Etiquetas de formato

** ... ** Coloca alineados al centro de la página todos los elementos que se coloquen entre las etiquetas.

…, **...**, **<u>...</u>**, **<tt>…</tt>** Dan atributos al texto que está encerrado dentro de las etiquetas, empleando ITALICAS, NEGRITAS, SUBRAYADO y MONOESPACIADO respectivamente.

Manejo de colores

Los colores se establecen con un número hexadecimal (0-F) de 6 dígitos, de los cuales los dos primeros representan la cantidad de rojo, el tercero y cuarto la cantidad de verde, y los dos últimos la cantidad de azul. Por ejemplo: #FFFFFF (Blanco), #FF0000 (Rojo), #00FF00 (Verde), #0000FF (Azul), #FFFF00 (Amarillo), #00FFFF (Cian), #FF00FF (Magenta), etc.

Colores de la página

Los colores de la página se pueden definir en la etiqueta de <body> tal y como se ve en el ejemplo siguiente:

<body text="#FFFFFF" link="#FF00FF" vlink="#FFFF00" alink="#FFFFFF" bgcolor="#000000">

Donde text define el color del texto de la página, link define el color de los vínculos nuevos, vlink el color de los vínculos ya visitados, alink el color de los vínculos que se están accesando en ese momento y bgcolor define el color del fondo o backgroud de la página.

Imagen como fondo de la página

La página puede tener como fondo una imagen en formato GIF, de tal forma que incluyendo el modificador background se puede incluir la imagen como mosaico en el fondo de la página. La etiqueta se emplea entonces como se ve en el ejemplo siguiente:

<body background="fondo.gif" >

Hiperligas

Las hiperligas permiten definir vínculos para hacer referencias o "brincar" hacia otras páginas y servicios de Internet. La estructura de una liga de hipertexto es como sigue:

 Texto o imagen del vínculo

Este tipo de ligas puede acceder a recursos como otra página Web, correo electrónico, ftp, gopher, usenet, telnet y otros. Por ejemplo se puede hacer referencia a estructuras como las siguientes.

WWW de mascarones
 Gopher de la UNAM

Anclas

Las anclas permiten definir vínculos o saltos internos entre diferentes partes de una misma página, esto es muy útil cuando la página es muy larga ya que permite ir rápidamente a los temas de más interés. Para esto, se define la posición a donde se va a brincar o mover rápidamente con una etiqueta como la siguiente: **Ancla 1**

Esta etiqueta se coloca en el lugar donde se posicionará el visualizador, al dar clic en una etiqueta como la siguiente: **<a href="#NOMBREANCLA"> Ir a la Sección uno</A>**

Listas de elementos con HTML

Las listas en HTML permiten elaborar rápidamente relaciones de elementos que son mostrados en diferentes formatos en las páginas de Web dependiendo del tipo de lista que se emplea.

Listas numeradas. Estas listas van numerando automáticamente los elementos conforme se van agregando.

```
<ol  type="1">
        <li>    Primer elemento
        <li>    Segundo elemento
        <li>    Hasta N elementos.
</ol>
```

Listas con viñetas. Estas listas colocan una marca o viñeta en cada uno de los elementos definidos, de tal forma que para agregar un nuevo elemento basta con colocar una nueva etiqueta <li> antes de cerrar la lista con y, seguido de la etiqueta, el texto que se quiere mostrar como nuevo elemento de la lista.

```
<ul>
        <li>    Primer elemento
        <li>    Segundo elemento
</ul>
```

Listas de definición. Las listas de definición están diseñadas para elaborar listas en las que se tenga que mencionar elementos acompañados de una descripción o texto alusivo, de tal forma que se abre la tabla con <dl> y se escriben los elementos o términos con la etiqueta <dt> y la definición que le acompaña con la etiqueta <dd>, y así para cada elemento de la lista.

```
<dl>
   <dt>         Término
      <dd>      Definición
   <dt>         Otro término
      <dd>      Otra definición
</dl>
```

Imágenes

La etiqueta **<img src="IMAGEN">** permite incorporar imágenes a una página Web, éstas deben estar en formato GIF o JPG ya que son los únicos formatos que reconoce el visualizador. Una imagen se inserta escribiendo una etiqueta HTML como la siguiente:

<img src="Archivo.gif"> ó <img src="Archivo.jpg">

Es posible cambiar el tamaño de la imagen empleando los modificadores de esta etiqueta width y height que se definen en pixeles o por porcentaje como se ve en el siguiente ejemplo:

<img src="Archivo.gif" width="200" height="200"> ó <img src="Archivo.gif" width="90%">

También se puede definir la alineación de una imagen respecto al texto escrito a continuación, de tal forma que se puede tener alineado en la parte superior (top), la parte inferior (bottom) o al centro (middle). <img src="Archivo.gif" align=top | bottom | middle>

Además, en el caso de los visualizadores que no pueden desplegar imágenes, se tiene un modificador (alt) para desplegar mensajes de texto alternos como se ve en el siguiente ejemplo:

<img src="Archivo.gif" alt="Texto Opcional">

Tablas

Las Tablas permiten incorporar a las páginas de Web las estructuras compuestas por renglones y columnas que tradicionalmente se manejan en los procesadores de texto. Para formar una tabla se escribe una estructura base como la siguiente:

```
<table border>
   <tr>
      <td>  PRIMER RENGLON, PRIMER COLUMNA  </td>
      <td>  PRIMER RENGLON, SEGUNDA COLUMNA  </td>
   </tr>
   <tr>
      <td>  SEGUNDO RENGLON, PRIMER COLUMNA  </td>
      <td>  SEGUNDO RENGLON,  SEGUNDA COLUMNA  </td>
    </tr>
</table>
```

La Tabla anterior es de dos renglones por dos columnas y en lugar de los textos escritos entre las etiquetas <td>...</td> se escribe el contenido que el usuario desea que aparezca en los cuadros (o celdas) de la tabla. El modificador border que va en la etiqueta <table> determina que la tabla lleva dibujadas las líneas divisorias y bordes de la tabla, por lo que si se deseara la tabla sin borde basta con eliminar este modificador. La tabla tendrá tantas columnas como etiquetas <td> escriba el usuario dentro de las <tr> y tantos renglones como etiquetas <tr> escriba dentro de las etiquetas <table>...</table>.

Formularios

Los formularios emplean una serie de instrucciones que definen acciones o elementos dentro del formulario tales como las siguientes etiquetas de HTML:

```
<form  method="post"  action="../cgi-bin/programas/datos">
        Contenido de la forma
</form>
```

La etiqueta <form> define el inicio de la forma y lleva los modificadores method, que puede ser "post" o "get" según el método de envío al servidor, así como action que indica el nombre del programa a ejecutar para procesar la información enviada por el formulario.

<input type="…" name="…"> La etiqueta <input> indica un elemento del formulario, donde type define la clase de elementos tales como "text" para campos de captura de texto junto con size=# donde # define el número de caracteres, "radio" para casillas de opciones únicas, "check" para casillas de verdadero o falso, "submit" para botones junto con value="texto" que define el texto que aparece en el botón, "reset" que genera un botón que borra los valores introducidos y regresa a los valores por default y que junto con value="texto" indica el nombre del botón. El atributo name indica el nombre del elemento.

Elementos básicos de un formulario

<input type="text" name="nombre" size="25" value="ESC. AQUI"> Esta etiqueta crea un campo para que se capture información mediante el teclado, asignando el contenido de éste a una variable "nombre" de tamaño 25 y con valor inicial "ESC. AQUÍ".

<input type="password" name="passw" size="10"> Esta etiqueta crea un campo para que se capture información mediante el teclado, ocultándola al escribirla, asignando el contenido de éste a una variable "passw" de tamaño 10.

<input type="radio" name="opciones" value="uno"> Esta etiqueta crea un campo de selección de opción única y se escribe con una lista de opciones, permitiendo seleccionar sólo una, asignando el contenido de este campo a una variable "opciones" y dándole el valor de "uno".

<input type="checkbox" name="multi" value="azul"> Esta etiqueta crea un campo de selección de opciones múltiples y permite seleccionar varias a la vez, asignándole el contenido de este campo a una variable "multi" dándole el valor de "azul".

<input type="submit" name="boton" value="Registrar"> Esta etiqueta crea un botón en el formulario de nombre "boton" y con el texto "Registrar".

<input type="reset" name="borrar" value="Borrar datos">
Esta etiqueta crea un botón en el formulario de nombre "borrar" y con el texto "Borrar datos".

Menús desplegables en un formulario

Es posible crear listas de opciones que aparecen en menús desplegables, para esto se utiliza la etiqueta <select> que define el nombre de la variable con "name" y las opciones se escriben con la etiqueta <option>, de tal forma que para crear una lista con los días de la semana se puede seguir el ejemplo siguiente:

```
<select name="dia">
        <option> Lunes
        <option> Martes
        <option> Miércoles
        <option> Jueves
        <option> Viernes
</select>
```

Áreas de texto en un formulario

Los campos de área de texto pueden contener varias líneas de texto a diferencia de los campos de tipo "text"; por lo que cuando la entrada es una serie de líneas continúas son muy útiles. La etiqueta empleada para incluir estos elementos se llama <textarea> y se escribe como sigue:

```
<textarea name="comentarios" rows="20" cols="20">
        Escriba aquí sus comentarios
</textarea>
```

La etiqueta anterior define un área de texto de 20 renglones por 20 columnas y el contenido de este campo se lo asigna a una variable de nombre "comentarios".

Frames

Los Frames son divisiones en forma de ventanas que se pueden definir desde HTML con la etiqueta <frameset>. Actualmente la mayoría de los visualizadores permiten la incorporación de frames, sin embargo, se debe tener presente que todavía existen algunos que no cuentan con esta capacidad por lo que se debe incluir la etiqueta <noframes> para poder considerar a estos visualizadores.

Para Incluir frames en una página se sustituye el <body> y se trabaja con <frameset>, de tal forma que se pueden crear frames horizontales o verticales dependiendo si se usa <frameset cols=“”> o <frameset rows=“”>. En cada frame se puede incorporar una página de WWW de tal forma que se emplea la etiqueta <frame src=“archivo.htm” name=“”> para incorporar el contenido de cada frame.

Como ejemplo de una página con “frames” se puede considerar el siguiente código de HTML:

```
<html>
        <frameset cols=“30%,70%”>
                <frame name=“uno” src=“primera.htm”>
                <frame name=“dos” src=“dos.htm”>
        </frameset>
</html>
```

En el caso de ligas de hipertexto se puede incorporar la etiqueta target=“nombreframe” para que se despliegue el contenido en el frame especificado en target.

```
<a href=“página.htm” target=“uno”>página uno </a>
```

Un ejemplo final de HTML se muestra en la figura 2.18.

Fig. 2.18. Formulario HTML.

2.4 Módulo IV. Administración de servidores de WWW con Linux (Apache)

Ahora que sabemos crear páginas Web, será necesario mostrarlas en Internet a través de un servidor Web. Apache es el servidor Web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma.

Tiene capacidad para servir páginas tanto de contenido estático como de dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información.

Un servidor WWW ofrece servicios (comunicación) dentro del World Wide Web en Internet mediante el protocolo HTTP (figura 2.19).

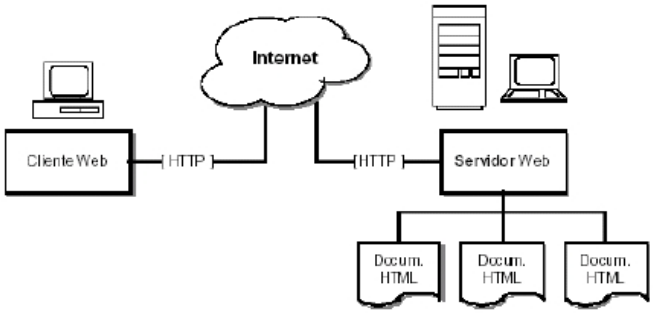


Fig. 2.19. El servidor Web.

HTTP es el protocolo de red para el WWW, basa su operación en la arquitectura Cliente-Servidor. El servidor HTTP es el encargado de publicar "recursos" electrónicos y el cliente consulta los recursos que el servidor ofrece.

Algunos criterios de selección de un servidor Web

- Función del Servidor de Web.
- Número de conexiones concurrentes.
- Número transacciones por segundo.
- Costo computacional por transacción.
- Proyección del crecimiento esperado.
- Soporte para la tecnología utilizada para el desarrollo.

Características de Apache

- Robusto, Soporte de un gran número de transacciones.
- Configurable para diferentes entornos de trabajo.
- Con un alto nivel de seguridad.
- Disponible para una gran variedad de plataformas.
- Soporte para servicio de proxy.
- Soporte para granjas de servidores.
- Soporte para lenguajes de script integrados como módulos (por ejemplo, PHP, mod_perl).
- Incluye el código fuente del servidor.
- Soporte para accesos restringidos.

- Soporte para SSL.
- Y además es gratuito.

Se obtiene de http://www.apache.org y su instalación se puede realizar de dos formas:

1. rpm -iv apache-1.3.27.rpm

2. tar zxvf apache-1.3.27.tgz
   ./configure
   make
   make install
   ./apachectl start → Para levantar el servidor.
   ./apachectl stop → Para detenerlo.

Configuración básica del servidor Web Apache

Su archivo de configuración se llama **httpd.conf**.

Directivas. Apache es administrado por más de 200 directivas las cuales permiten que determinada funcionalidad pueda ser incluida. En Linux, el administrador controla qué directivas estarán disponibles de acuerdo a los módulos con los que se compila Apache. La configuración se realiza mediante directivas organizadas en tres grupos:

Global Environment. Esta sección administra las directivas generales de operación para apache.
Main Server. Administra las directivas del servidor principal o estándar de Apache.
Virtual Host. Administra las directivas donde los mismos procesos de Apache soportan diversos nombres de dominio.

Global Environment

ServerName. Define el nombre para el servidor Apache, el cual será utilizado al construir los URLs cuando el cliente solicite otros recursos Web (páginas, phps, cgis, etc.) del servidor. Esta etiqueta se utiliza tanto en ServerConfig como en VirtualHost.

User. Esta directiva define el user ID mediante el cual Apache operará. Valor por Default: User #-1.

Group. Define el group ID mediante el cual Apache operará. Valor por Default: User #-1.

Port (Reemplazado por **Listen** en 2.0). Esta directiva define cuál es el puerto en que operará el servidor de Web. Valor por Default: 80.

ServerAdmin. Esta directiva define el correo electrónico del administrador del servidor Web e indica la dirección a incluirse en los mensajes de error que serán enviados al cliente. Es necesario tener habilitada la directiva ServerSignature Email.

DocumentRoot. Esta directiva define la ruta absoluta donde se almacenarán los archivos HTML que se desean publicar. Valores por Default:

Compilado: /usr/local/apache/htdocs
Instalado : /var/www/html

Esta etiqueta se utiliza tanto en ServerConfig como en VirtualHost.

ServerRoot. Esta directiva define la ruta absoluta donde los directorios “conf” y “logs” podrán ser encontrados. Valor por default: /usr/local/apache. Se utiliza tanto en ServerConfig como en VirtualHost.

MinSpareServers. Define cuál es el número mínimo de procesos servidores que son mantenidos en espera. Se monitorea el número de conexiones en espera, si ésta es menor a “MinSpareServers” se levantan los demonios necesarios. Valor por default: 5.

MaxSpareServers. Esta directiva define cuál es el número máximo de procesos servidores que son mantenidos en espera. Se monitorea el número de conexiones en espera, si ésta es mayor a “MaxSpareServers” se eliminan los demonios necesarios, siempre y cuando no estén realizando alguna tarea.

StartServers. Define cuál es el número máximo de servidores que se iniciarán cuando se arranca Apache. Valor por default: 5.

MaxClients. Esta directiva define cuál es el número máximo de procesos servidores que como máximo podrán ser iniciados, con el objeto de atender las solicitudes de los clientes. Valor por default: 20.

ErrorDocument. En caso de que un error o problema ocurra con Apache cuando se solicite un recurso, éste puede ser configurado para que haga una de las siguientes acciones:

- Enviar un mensaje de error (default).
- Enviar un mensaje personalizado.
- Redirigir a un URL local para manejar el problema o error.
- Redirigir a un URL externo quien manejará el problema o error.

Ejemplos de ErrorDocument:

ErrorDocument	500	http://foo.example.com/cgi-bin/tester
ErrorDocument	404	/cgi-bin/bad_urls.pl
ErrorDocument	401	/subscription_info.html
ErrorDocument	403	“Lo sentimos, el servicio no esta disponible”

HTTP 1.1 (Códigos de estado)

- 200 OK.
- 102 Processing.
- 204 No content.
- 400 Bad Request.
- 404 Not Found.
- 403 Forbidden.
- 405 Method not allowed.
- 505 HTTP version not supported.
- 507 Insufficient storage.

PidFile. Define el lugar donde se almacenará el archivo “PidFile”, el cual contiene el PID que es un valor numérico que identifica al proceso httpd “padre”. Si este archivo se pierde es común tener problemas cuando arranca o se detiene Apache vía el script en /etc.

PidFile file
Default file: logs/httpd.pid

Bitácoras – Logging. Apache cuenta con dos archivos donde residen las bitácoras:

access.log. Registra todos los accesos al sitio.
error.log. Registra los errores que genere un acceso al sitio o que los procesos de Apache reporten.

ErrorLog <filename>. Esta directiva define el nombre del archivo en el cual el servidor registrará los errores que se presenten. Si <filename> comienza con un "/" se considera que la ruta depende de la raíz del "Filesystem", si no, se considera parte del ServerRoot.

Loglevel <error>. Esta directiva define el nivel de mensajes de error que serán registrados en la bitácora de "ErrorLog" (emerg, alert, crit, error).

CustomLog <filename>. Define el nombre del archivo en el cual el servidor registrará los accesos que se presenten. Si <filename> comienza con un "/" se considera que la ruta depende de la raíz del "Filesystem", si no, se considera parte del ServerRoot.

LogFormat "format-string" <nickname>. "Format-string": "%h %l %u %t \"%r\" %>s %b"

%h = Registra la ip del cliente, hostname.
%l = Si el deamon identd corre en el cliente, reporta la información que el identd devuelva.
%u = Si se requiere de un username y password para acceder, en este campo se registra.
%t = Fecha y hora del request en el formato [dia/mes/año:hora:minuto:segundo tzofffset].
\"%r\"= Recurso solicitado por el cliente, entre comillas, request.
%>s = Un código de tres dígitos donde se muestra el valor devuelto al cliente, status.
%b = El número de bytes devueltos exceptuando encabezados.

Existe una serie de LogFormat preestablecidos.

- **Common** Detallado anteriormente.
- **Refered** Orientado a llevar un registro de las rutas de los usuarios dentro del sitio.
- **Agent** Orientado a registrar el nombre de los navegadores de los clientes.
- **Combined** Combina las características de los anteriores.

HostNameLookup [on|off]. Esta directiva habilita la resolución de nombres en el DNS cuando se establece una conexión. Por razones de rendimiento se sugiere no habilitarla. Para la resolución de nombres en las bitácoras existe el programa logresolve, también de Apache, que puede realizar la resolución de nombre fuera de línea.

<Directory>. Permite aplicar otras directivas de forma específica a todos los recursos dentro de la ruta definida por "dir". Se deben considerar rutas absolutas.

```
<Directory /usr/local/apache/htdocs>
        ...
        …
</Directory>
```

Options. Esta directiva controla qué características del servidor están disponibles para un directorio en particular. Options [+|-] option [[+|-]option]. Puede ser colocado a "option none" en caso de que no se desee ninguna funcionalidad adicional.

Los valores que puede tomar **option** son los siguientes:

All. Todas las opciones se activan excepto MultiViews, (Esta es la opción por default).

ExecCGI. La ejecución de scripts CGIs es permitida.

FollowSymLinks. Las ligas simbólicas son validas dentro de este directorio. Aún cuando el servidor seguirá las ligas simbólicas éste no cambiará de la ruta definida por la directiva <Directory>.

Includes. Permite el uso de Server-side includes.

IncludesNOEXEC. Server-side includes son permitidos, pero #execCGI son desactivados.

Indexes. Si un URL mapea a un directorio solicitado y no hay DirectoryIndex (index.html) en este directorio, entonces el servidor devolverá un listado de directorio. Ejemplos:

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

<Directory /web/docs/spec>
    Options Includes
</Directory>

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

Sitios Web Dinámicos: PHP, JSP, Servlets, ASP, ColdFusion, ActiveX, JavaScript, Applets, CGIs.

CGIs. La tecnología CGI "Common Gateway Interface" define un modelo de programación que puede ser implementado por múltiples lenguajes. Los CGIs son programas que siguen un estándar definido, corren en el servidor, reciben parámetros desde el cliente y su salida es enviada al navegador. Fueron la primera alternativa para generar dinamismo a un sitio Web.

ScriptAlias <alias> <path-filesystem> → ScriptAlias /cgi-bin /usr/www/cgi-bin

Convierte las solicitudes vía URL al path absoluto donde residen los programas CGI. Esta etiqueta se utiliza tanto en ServerConfig como en VirtualHost.

AddHandler cgi-scripts .cgi .pl. Permite que determinada extensión sea relacionada a un evento en particular. En el caso de los CGIs, lo que se indica es que las extensiones .cgi y .pl quedan identificadas como un script. Apache las interpretará en lugar de sólo enviar el contenido del archivo al cliente.

Server-Side Includes. La opción SSI activa un filtro que permite incluir etiquetas dentro de un archivo HTML, las cuales son procesadas por Apache antes de ser enviadas como HTML al cliente. Es necesario habilitar la opción +Includes en el directorio donde se encuentran los archivos que incluyen etiquetas SSI.

Directivas requeridas:

AddType text/html .shtml. Relaciona los archivos con extensión shtml como aquellos que contienen etiquetas SSI.

AddOutputFilter INCLUDES .shtml. Habilita el filtro para los archivos .shtml.

"SSI representa un riesgo y debe ser usado con cautela, para evitar un incidente de seguridad"

VirtualHost. Los servidores virtuales permiten que un mismo servidor Apache pueda responder a diferentes solicitudes, con lo cual es posible mantener múltiples sitios Web con diferentes nombres y/o direcciones IP.

```
<VirtualHost servername>
</VirtualHost>
```

```
NameVirtualHost 192.168.40.5
<VirtualHost 192.168.40.5>
   ServerAdmin root@web.ejemplo.com
   ServerName www.micasa.com
   DocumentRoot /home/micasa/htdocs/
</VirtualHost>

<VirtualHost 192.168.40.5>
   ServerAdmin root@web.ejemplo.com
   ServerName www.miempresa.com
   DocumentRoot /home/miempresa/htdocs/
</VirtualHost>
```

Control de Acceso. Apache ofrece la funcionalidad para control de acceso, con lo cual es factible determinar las direcciones IP y los usuarios que tendrán acceso sobre recursos específicos del servidor Web. La configuración de acceso vía nombre de dominio o vía IP puede realizarse con ayuda de las siguientes directivas:

Order. Define el orden en que las directivas allow o deny serán implementadas.
Allow. Define los hosts que tendrán acceso al recurso.
Deny. Define los hosts que no tendrán acceso al recurso.

```
Allow from all|host|env=env-variable [host|env=env-variable]
Deny from all|host|env=env-variable [host|env=env-variable]
```

Ejemplo:

```
<Directory /usr/local/apache/htdocs/intranet >
        order allow,deny
       Allow from 192.168.0.0/255.255.255.0
       Deny from  all
</Directory>
```

Con Apache también es factible determinar un login y password por usuario, quienes tendrán acceso sobre recursos específicos del servidor Web.

AuthUserFile <archivo de autenticación>. Indica cuál es el archivo que contiene la lista de usuarios y passwords para realizar la autenticación. Es importante que este archivo no sea accesible por los clientes Web.

Ejemplo: /usr/local/apache/conf/.htpasswd

AuthGroupFile <archivo de autenticación>. Define cuál es el archivo que contiene la lista de grupos y los usuarios que la conforman para realizar la autenticación. Ejemplo /dev/null

AuthName "banner-string". Define el nombre de autorización para el recurso protegido, éste aparecerá como un mensaje en la caja de dialogo que solicita el password para acceder al recurso protegido. Ejemplo: "Nombre del recurso protegido".

require valid-user. Esta directiva indica que se requiere de un usuario y una clave de acceso para un recurso determinado.

Ejemplo de autenticación:

```
<Directory /usr/apache/htdocs/intranet >
      AuthUserFile /usr/apache/conf/claves/.htpasswd
      AuthGroupFile /dev/null
      AuthName "Bienvenidos a la Intranet"
      AuthType Basic
      require valid-user
</Directory>
```

htpasswd –C /usr/apache/conf/claves/.htpasswd <username>

Módulos. En Apache los módulos son los encargados de agregar funcionalidad adicional incrementando el número de directivas disponibles.

- **Módulos estáticos**. Se agregan al momento de compilar Apache.
- **Módulos dinámicos**. Se agregan durante el ambiente productivo, permiten adicionar funcionalidad sin necesidad de recompilar Apache.

2.5 *Módulo V. Programación con PHP*

Continuando con la preparación de nuestro ambiente de desarrollo (servidor Web, servidor de bases de datos y un intérprete) es el momento de ocuparnos de PHP.

Aplicación Web

Una aplicación Web consiste básicamente en generar documentos HTML dinámicos, por dinámicos se debe entender que dichos documentos de HTML no existen como tal en el servidor Web, sino que se generan en el momento en que son solicitados.

El concepto en si es sencillo, como podemos ver el protocolo es el siguiente (figura 2.20): el servidor Web está en espera de una petición que el cliente genera por medio de un navegador Web, una vez que se presenta, el servidor Web se encarga de procesar la información, generar un documento de HTML y devolver el documento de HTML al navegador.

Entrando en detalle acerca del proceso que se realiza dentro del servidor Web, una vez que existe una petición éste debe contar con un intérprete, que es el que realmente atiende la petición del cliente. Este intérprete es el encargado de generar el documento HTML y de realizar consultas a un servidor de Base de Datos (DBMS), cuando el DBMS atiende la consulta del intérprete la procesa y devuelve un record set, que no es más que un área de memoria donde se colocan los resultados de la consulta. Después, el intérprete continúa con el proceso de generación de la pagina Web y una vez terminado se la entrega al servidor Web, que la envía al cliente que realizó dicha petición.

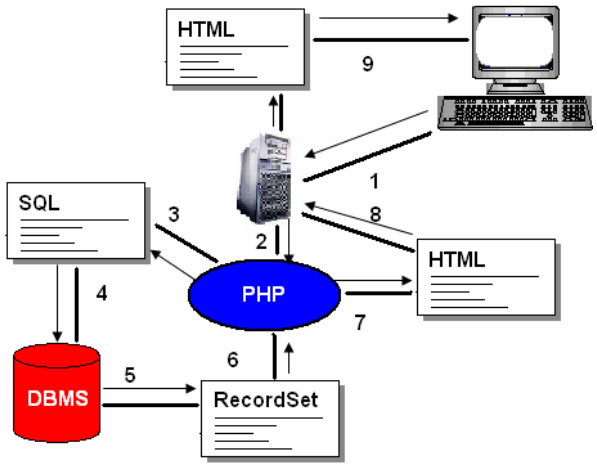


Fig. 2.20. Una aplicación Web.

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos o enviar y recibir cookies.

Es muy utilizado para hacer scripts del lado del servidor. Se necesitan tres cosas para que esto funcione: El intérprete PHP, un servidor Web y un navegador. Es necesario correr el servidor Web con PHP instalado. El resultado del programa se puede obtener a través del navegador, conectándose con el servidor Web.

Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si lo conocemos bien y quisiéramos utilizar algunas características avanzadas en programas clientes, podemos utilizar PHP-GTK para escribir dichos programas, es una extensión de PHP no disponible en la distribución principal.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux; soporta la mayoría de servidores Web de hoy en día, incluyendo Apache.

De modo que, con PHP se tiene la libertad de elegir el sistema operativo y el servidor. También se tiene la posibilidad de usar programación procedimental o programación orientada a objetos. Aunque no todas las características estándar de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas bibliotecas y aplicaciones grandes están escritas íntegramente usando este paradigma de programación.

Con PHP no estamos limitados a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash (usando libswf y Ming) sobre la marcha. También puede presentar otros resultados, como XHTM y archivos XML. PHP puede auto generar estos archivos y almacenarlos en vez de presentarlos en la pantalla.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Cuenta con muchas otras extensiones y funciones muy interesantes que se pueden ver en la documentación.

En términos generales, las cosas para tener en cuenta en un lenguaje de script son: velocidad, estabilidad, seguridad y simplicidad.

- **Velocidad:** No sólo la de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debemos requerir demasiados recursos de sistema.

- **Estabilidad:** La velocidad no sirve de mucho si el sistema se cae cada cierta cantidad de ejecuciones. Ninguna aplicación es 100% libre de bugs, pero teniendo de respaldo una increíble comunidad de programadores y usuarios es mucho más difícil para lo bugs sobrevivir. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

- **Seguridad:** El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo .ini.

- **Simplicidad:** Se le debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente (figura 2.21).

Lenguaje de programación PHP

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor.

```
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>
      <?php
         echo "Hola ; soy un script PHP!";
      ?>
    </body>
</html>
```

Fig. 2.21. La simplicidad de PHP.

Lo que distingue a PHP de la tecnología JavaScript, la cual se ejecuta en la máquina cliente, es que el código es ejecutado en el servidor. Si tuviésemos un script similar al de la figura 2.21, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido.

Instalación

Descargar. Podemos descargar PHP desde la sección de descargas de su sitio Web (http://www.php.net) el cual tiene varios mirrors. Para la mayoría de los usuarios de PHP que tienen sistemas tipo Unix se recomienda que descarguen y compilen el código fuente.

Descomprimir. Extraer el código fuente del archivo .tar.gz que acabamos de descargar es fácil:

$tar –zxvf php-xx.tar.gz

Este comando creara un nuevo directorio dentro del que actualmente nos encontramos y que contendrá el código fuente de la distribución. Debemos cambiarnos a ese directorio con **cd** para proceder a compilar el servidor PHP.

Configuración. Es sumamente sencilla ya que sólo tenemos que ejecutar el siguiente comando:

$./configure –prefix=/usr/local/php
-apxs=/usr/local/apache/bin/apxs
-with-mysql=/usr/local/mysql
-with-postgresql=/usr/local/postgresql

Como podemos observar el prefix es la ruta en la que se instalará PHP.

Compilación. Ahora podemos compilar las diferentes partes que forman PHP simplemente ejecutando el siguiente comando:

$make

Puede tardar varios minutos en compilar, este tiempo varía considerablemente en función del hardware.

Instalar. Ejecutar el comando:

$make install

Sintaxis básica

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (**<?php. . .?>** y **<script language="php">. . .</script>**) están siempre disponibles; el resto pueden ser configuradas en el fichero php.ini para ser o no aceptadas por el intérprete.

El primer método, <?php. . .?>, es el más conveniente, ya que permite el uso de PHP en código XML como XHTML.

Variables

En PHP las variables se representan como un signo de pesos ($), seguido por el nombre de la variable. Éste es sensible a minúsculas y mayúsculas, por lo que no es lo mismo $myvar que $Myvar.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Tiene que empezar con una letra o una raya (underscore), seguido de cualquier número de letras, números y rayas (figura 2.22).

```
<?php
    $var = "David";
    $Var = "García";
    echo "$var, $Var";     // escribe "David, García"

    $4site = '127.0.0.1';  // no puede empezar con numero
    $_4site = '127.0.0.1'; // puede empezar con guión bajo
?>
```

Fig. 2.22. Las Variables en PHP.

Tipos de datos

El tipo de una variable usualmente no es declarado por el programador; en cambio, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usada. PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

- Boolean.
- Integer.
- Float(número de punto-flotante, también conocido como 'double').
- String.

Dos tipos compuestos:

- Array.
- Object.

Y finalmente dos tipos especiales:

- Resource. Es una variable que contiene una referencia a un recurso externo. Los recursos son creados y usados por funciones especiales.
- NULL. Representa que una variable no tiene valor.

Expresiones

En PHP, casi cualquier cosa que escribimos es una expresión. La forma más simple y ajustada de definir una expresión es "cualquier cosa que tiene un valor".

Las formas más básicas de expresiones son las constantes y las variables. Cuando escribimos "$a=5", estamos asignando '5' a $a. En este caso, '5' es una constante entera.

Después de esta asignación, se espera que el valor de $a sea 5 también, de manera que si escribes $b=$a, se espera también que se comporte igual que si escribiésemos $b=5. En otras palabras, $b es una expresión también con el valor 5. Si todo va bien, eso es exactamente lo que pasará. Las funciones son un ejemplo algo más complejo de expresiones.

Otro buen ejemplo de expresiones es el pre y post incremento y decremento (variable++ y variable--). Los operadores aritméticos se muestran en la tabla 2.3.

Las expresiones de comparación se evalúan a 0 o 1, significando falso (**FALSE**) o verdadero (**TRUE**), respectivamente. PHP soporta los operadores relacionales y lógicos mostrados en la tabla 2.4. Se usan frecuentemente dentro de la ejecución condicional como la instrucción if. Hay muchas más expresiones.

Operadores

Operador	Acción
-	Resta binaria, también menos unario
+	Suma
*	Multiplicación
/	División
%	Módulo(residuo)
--	Decremento
++	Incremento

Tabla 2.3. Operadores aritméticos.

Operador	Acción
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
==	Igual
!=	Diferente

Tabla 2.4. Operadores relacionales y lógicos.

Bloques y sentencias

Además de los bloques ya conocidos (**if**, **while**, **do**/**while**, **for**), PHP cuenta con el ciclo **foreach** (figura 2.23) que nos ayuda a recorrer los valores de un array, lo cual puede resultar muy útil por ejemplo para efectuar una lectura rápida del mismo. Recordemos que un array es una variable que guarda un conjunto de elementos (valores) catalogados por claves.

```
foreach ($array as $clave=>$valor)
{
   instruccion1;
   instruccion2;
   ……;
}
```

Fig. 2.23. Ciclo foreach.

Break y continue

Estas dos instrucciones se introducen dentro de una estructura y nos sirven respectivamente para escapar del ciclo o saltar a la iteración siguiente. Pueden resultarnos muy prácticas en algunas situaciones.

Funciones

PHP proporciona un conjunto de funciones de gran utilidad, algunas de ellas se muestran en la tabla 2.5.

Función	Descripción de la Función
chr(n)	Devuelve el carácter cuyo código ASCII es n.
ord(cadena)	Devuelve el código ASCII del primer carácter de la cadena.
strlen(cadena)	Devuelve la longitud (nº de caracteres incluidos los espacios) de la cadena.
strtolower(cadena)	Cambia los caracteres de la cadena a minúsculas.
strtoupper(cadena)	Convierte en mayúsculas todos los caracteres de la cadena.
Ucwords(cadena)	Convierte a mayúsculas la primer letra de cada palabra.
ucfirst(cadena)	Convierte a mayúsculas la primer letra de la cadena y pone las restante en minúsculas.
ltrim(cadena)	Elimina los espacios al principio de la cadena.
rtrim(cadena)	Elimina los espacios al final de la cadena.
trim(cadena)	Elimina los espacios tanto al principio como al final de la cadena.
chop(cadena)	Elimina los espacios al final de la cadena. Es un alias de rtrim.
substr(cadena,n)	Si el valor de n es positivo extrae todos los caracteres de la cadena a partir del que ocupa la posición enésima a la izquierda. Si n es negativo extrae los n últimos caracteres de la cadena.
substr(cadena,n,m)	Si n y m son positivos, extrae m caracteres a partir del que ocupa la posición enésima contados de izquierda a derecha. Si n es negativo y m es positivo, extrae m (contados de izquierda a derecha) a partir del que ocupa la posición enésima contada de derecha a izquierda. Si n es positivo y m es negativo, extrae la cadena comprendida entre el enésimo carácter (contados de izquierda a derecha) hasta el enésimo contado de derecha a izquierda. Si n es negativo y m también es negativo, extrae la cadena comprendida entre el enésimo y el enésimo caracteres contado de derecha a izquierda. Si el valor absoluto de n es menor que el de m devuelve una cadena vacía.
strrev(cadena)	Devuelve la cadena invertida.
str_repeat(cadena, n)	Devuelve la cadena repetida tantas veces como indica n.

Tabla 2.5. Algunas funciones de PHP.

Constantes

Se puede definir una constante usando la función **define()** (figura 2.24). Una vez definida no puede ser modificada ni eliminada. Sólo se puede definir como constantes valores escalares (**boolean**, **integer**, **float** y **string**). Para obtener el valor de una constante sólo es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo $. También se puede utilizar la función **constant()** para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica se usa la función **get_defined_constants()** parar obtener una lista de todas las constantes definidas.

```
<?php
    define("CONSTANT", "Hola Mundo.");
    echo CONSTANT; // escribe "Hello world."
?>
```

Fig. 2.24. Definiendo una constante.

Herramientas elementales

array_shift(). Esta función extrae el primer elemento del array y lo devuelve. Además, acorta la longitud del array eliminando el elemento que estaba en la primera casilla.

unset(). Se utiliza para destruir una variable dada. En el caso de los arreglos, se puede utilizar para eliminar una casilla de un array asociativo (los que no tienen índices numéricos sino que su índice es una cadena de caracteres).

array_push(). Inserta al final del array una serie de casillas que se le indiquen por parámetro.

array_merge(). Une dos arrays. A ésta se le pasan dos o más arrays por parámetro y devuelve un arreglo con todos los campos de los vectores pasados.

PHP también tiene opciones que permiten crear, modificar, leer y obtener información sobre **archivos** de cualquier tipo.

Manejo de cadenas

strlen(string cad). Devuelve la longitud de la cadena.

str_split. Convierte una cadena en una matriz. Si el parámetro opcional longitud_separacion es especificado, la matriz devuelta estará separada en pedazos, cada uno de longitud longitud_separacion, de otro modo cada trozo tendrá un carácter de longitud.

str_word_count. Cuenta el número de palabras en cadena.

strcasecmp(string cad1, string cad2). Devuelve < 0 si cad1 es menor que cad2; > 0, si cad1 es mayor que cad2, y 0 si son iguales.

strtolower(string cad). Devuelve la cadena con todas sus letras en minúsculas.

strtoupper(stringcadena). Devuelve la cadena con todas sus letras en mayúsculas.

substr(string cadena, int comienzo [, int largo]). Devuelve la porción de cadena especificada por los parámetros comienzo y largo.

Entrada de datos a partir de un formulario

Todo formulario comienza con la etiqueta:

<form action="direccion.php" method="post/get">

Con **action** indicamos la dirección del script que va procesar la información que recogemos en el formulario, mientras que **method** nos indica la forma de envío de los datos, "post" o "get". La etiqueta **</form>** indica el final del formulario.

Dentro de nuestro formulario podemos usar diversos elementos HTML, input o select, cuyos valores son rellenados por el visitante de la página. Pues bien, al enviar el formulario al script indicado por el atributo **action**, automáticamente se genera una variable cuyo nombre es el especificado con el atributo **name** y cuyo valor es el introducido por el usuario.

La diferencia entre get y post radica en la forma de enviar los datos a la página, mientras que el método get envía los datos usando la URL, el método post los envía por la entrada estándar STDIO.

SESIONES

PHP nos permite almacenar variables llamadas “de sesión” que, una vez definidas, podrán ser utilizadas durante este periodo de tiempo por cualquiera de los scripts de nuestro sitio. Estas variables serán específicas del usuario de modo que varias variables de sesión del mismo tipo con distintos valores pueden estar coexistiendo para cada una de las sesiones que están teniendo lugar simultáneamente. Estas sesiones tienen además su propio identificador de sesión que será único y específico.

Para iniciar una sesión podemos hacerlo de dos formas distintas:

- Declaramos abiertamente la apertura de sesión por medio de la función session_start(). Ésta crea una nueva sesión para un nuevo visitante o bien recupera la que está siendo llevada a cabo.
- Declaramos una variable de sesión por medio de la función session_register('variable'). Ésta 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 de ser iniciadas al principio de nuestro script, antes de abrir cualquier etiqueta o de imprimir cualquier cosa. En caso contrario recibiremos un error.

Otras variables de sesión importantes son las que nos muestra la tabla 2.6.

Función	Descripción
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.

Tabla 2.6. Variables de sesión.

Acceso a las bases de datos MySQL con PHP

Para poder consultar, insertar, borrar o actualizar datos desde PHP con nuestro servidor MySQL es necesario primero establecer una conexión, para ello empleamos la función **mysql_connect** como lo muestra la figura 2.25.

```
<?php
    $link = mysql_connect('localhost', 'mysql');
    if (!$link) {
        die(' error de conexión : ' . mysql_error());
}
?>
```

Fig. 2.25. Conexión con MySQL.

Acceso a las bases de datos PostgreSQL con PHP

La conexión con PostgreSQL se lleva a cabo con la función pg_connect (figura 2.26).

```
<?php
    //Conexión con la base de datos
    $connection=pg_connect (“localhost”, “”, “”, “”, “ejemplo”);
?>
```

Fig. 2.26. Conexión con PostgreSQL.

2.6 Módulo VI. Interacción de WWW con bases de datos (MySQL)

MySQL es, hoy por hoy, uno de los sistemas gestores de base de datos más populares del mundo. Las razones son la sencillez de instalación y el manejo de su estabilidad, pero sobre todo su rapidez de funcionamiento y su costo. Lo desarrolla, distribuye y soporta MySQL AB, una compañía comercial fundada por los desarrolladores de MySQL. Es una compañía Open Source de segunda generación que une los valores y metodología Open Source con un exitoso modelo de negocio.

La parte SQL de "MySQL" se refiere a "Structured Query Language". SQL es el lenguaje estandarizado más común para acceder a bases de datos y está definido por el estándar ANSI/ISO SQL. Éste ha evolucionado desde 1986 y existen varias versiones.

Algunas características de MySQL

- Escrito en C y en C++.
- Funciona en diferentes plataformas.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Proporciona sistemas de almacenamiento transaccionales y no-transaccionales.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible.
- Diversos tipos de columnas: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM y tipos espaciales OpenGIS.
- Soporte completo para las cláusulas SQL GROUP BY y ORDER BY. Soporte de funciones de agrupación (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN() y GROUP_CONCAT()).
- Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN.
- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Instalación

Descargar. Podemos descargar MySQL desde la sección de descargas de su sitio web (http://www.mysql.com) el cual tiene varios mirrors. Para la mayoría de los usuarios de MySQL que tienen sistemas tipo Unix se recomienda que descarguen y compilen el código fuente. El proceso de compilación es sencillo y permite adaptar el servidor MySQL a nuestras necesidades.

Después de la descarga, es importante que verifiquemos que el archivo descargado está completo y sin modificaciones. Esto podemos hacerlo comparando el archivo descargado (.tar.gz) con su firma PGP.

Agregar Grupo. Antes de crear el usuario que será el administrador del servidor de MySQL es recomendado generar un grupo y, a partir de éste, agregar a los usuarios de MySQL. Para ello es necesario utilizar el comando:

shell> addgroup *mysql*

Agregar Usuario. Una vez generado el grupo, el siguiente paso es crear un usuario, el cual se agregará al grupo que creamos en el paso anterior. Para ello utilizamos el comando siguiente:

shell> useradd *–g mysql* mysql

Descomprimir. Para extraer el código fuente del archivo .tar.gz que acabamos de descargar ejecutamos el siguiente comando:

shell> tar –zxvf mysql-xx.tar.gz

Este comando creará un nuevo directorio dentro del que actualmente nos encontramos y que contendrá el código fuente de la distribución. Debemos cambiarnos a ese directorio con el comando **cd** para proceder a compilar el servidor MySQL.

Configuración. Sólo tenemos que ejecutar el siguiente comando:

shell> ./configure –prefix = /usr/local/mysql

Como podemos observar el prefix es la ruta en la que se instalará el servidor de MySQL.

Compilar. Ahora podemos compilar las diferentes partes que forman MySQL simplemente ejecutando el siguiente comando:

shell> make

Debemos tener un poco de paciencia, una configuración básica tarda varios minutos en compilar, pero este tiempo puede variar considerablemente en función del hardware.

Instalar. Ahora es el momento de instalar el paquete en el directorio elegido en prefix ejecutando:

shell> make install

Si se desea crear un fichero de opciones, debe utilizarse como plantilla uno de los presentes en el directorio support-files. Por ejemplo:

shell> cp support-files/my-medium.cnf /etc/my.cnf

- El primer paso es ubicarnos en el directorio donde se instalo el servidor:

shell> cd /usr/local/mysql

- Si no se ha instalado antes MySQL, se deben crear las tablas de permisos:

shell> bin/mysql_install_db --user=mysql

Si se ejecuta el comando como usuario root se debe emplear la opción --user tal como se muestra. El valor de la opción debe ser el nombre de la cuenta de usuario creada para permitir que el servidor se ejecute. Si se ejecuta el comando habiendo iniciado sesión como este último usuario, se puede omitir la opción --user.

Después de crear o actualizar la tabla de permisos mediante mysql_install_db, habrá que reiniciar el servidor manualmente.

Se debe cambiar el propietario de los programas binarios a root y el propietario del directorio de datos al que se creó para ejecutar mysqld. Asumiendo que se está en el directorio de instalación (/usr/local/mysql), los comandos serían similares a estos:

shell> chown -R root .
shell> chown -R mysql var
shell> chgrp -R mysql .

El primer comando cambia el atributo de propietario de los ficheros y les asigna el usuario root. El segundo cambia el atributo de propietario del directorio de datos y le asigna el usuario mysql. El tercero cambia el atributo de grupo, asignándolo al grupo mysql.

- El siguiente paso es levantar el servidor para ello utilizamos el comando:

shell> bin/mysqld_safe --user=mysql &

Nota: Si se comienza desde un RPM fuente, debe procederse así:

shell> rpmbuild --rebuild --clean MySQL-*VERSION*.src.rpm

Una vez que hemos levantado el servidor el siguiente paso es acceder a la consola de MySQL, para ello hay que utilizar el comando:

shell> mysql –u mysql

El cual nos devuelve un prompt como el siguiente:

mysql>

Ahora nos encontramos en la consola de MySQL, utilizando el comando **show databases;** podemos observar las bases de datos que existen actualmente en el servidor (figura 2.27).

mysql> show databases;

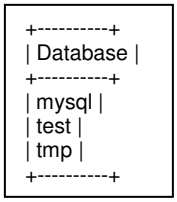


Fig. 2.27. Las bases de datos del servidor.

La base de datos mysql es necesaria porque es la que describe los privilegios de acceso de los usuarios, test se provee para que los usuarios hagan pruebas.

Para trabajar con una base de datos en particular utilizamos el comando **use NombreBD;** con el nombre de la base de datos que deseemos trabajar.

Para salir de la consola de MySQL sólo hay que utilizar el comando **exit**, el cual nos regresa el shell del sistema operativo.

Si deseamos dar de baja el servidor de MySQL utilizamos el comando:

$mysqladmin shutdown

El cual cerrara las transacciones existentes en el servidor así como las conexiones y lo dará de baja de forma segura.

Creación de bases de datos en MySQL

Para poder generar una base de datos requeriremos de acceder a la consola de MySQL. Una vez dentro, el siguiente paso es ejecutar el siguiente comando de SQL:

mysql> create database nuevabd;

Si nuestra query (consulta) no tiene errores mostrará un mensaje parecido al siguiente:

Query Ok, 1 row affected (0.002 sec)

Este mensaje quiere decir que nuestro query se ejecuto con éxito. Ahora nos podemos cambiar a la base de datos creada para empezar a generar nuestras tablas:

mysql> use nuevabd;

De esta manera la consola muestra un mensaje como el siguiente:

Database changed

Crear una tabla

La creación de la base de datos es una tarea sencilla, pero hasta ahora permanece vacía, como lo muestra el comando:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

La parte difícil es decidir cómo debería ser la estructura de la base de datos, qué tablas se necesitan y qué columnas habrá en cada tabla.

Debemos usar la sentencia **create table** para especificar la estructura de una tabla:

Sintaxis básica de CREATE TABLE:
CREATE TABLE [IF NOT EXISTS] *tbl_nombre*
[(*definición*,...)]
[*opciones*]

Para verificar que la tabla ha sido creada en la forma esperada, utilizamos la sentencia **describe**:

mysql> describe tbl_nombre;

Ésta puede ser utilizada en cualquier momento, por ejemplo, si olvidamos los nombres o el tipo de dato de las columnas de la tabla.

La sintaxis básica de inserción de datos es la siguiente:
INSERT
[INTO] *tbl_name* [(*col_name*,...)]
VALUES ({*expr* | DEFAULT},...),(...),...

Extraer información de una tabla. La sentencia SELECT es utilizada para traer información desde una tabla. La sintaxis general de esta sentencia es:

SELECT *seleccionar_esto*
FROM *desde_tabla*
WHERE *condiciones*;

- **seleccionar_esto** es lo que se quiere ver. Puede ser una lista de columnas ó "*" para indicar "todas las columnas.".
- **desde_tabla** indica la tabla donde están los datos a recuperar.
- La cláusula WHERE es opcional. Si está presente, **condiciones** representa las "condiciones" que cada registro debe cumplir para retornar como resultado.

Existen mucho más sentencias y funciones en MySQL cuyo funcionamiento se puede observar en la documentación.

Tipos de tabla

Al contrario de otros sistemas gestores de bases de datos, MySQL permite utilizar diferentes tipos de tabla, dependiendo de nuestras necesidades.

Si necesitamos velocidad de proceso podemos utilizar tablas que estarán directamente en la memoria RAM del servidor. Si la prioridad es ahorrar espacio de almacenamiento MySQL puede comprimir tablas para que ocupe lo menos posible. Es posible utilizar transacciones si se trata de uno de los requisitos de la aplicación.

Y lo mejor de todo, si sólo se necesita un sistema para almacenar datos y posteriormente consultarlos, sin quebraderos de cabeza, podemos olvidar que existen los diferentes tipos de tabla.

- MyISAM
- InnoDB
- HEAP
- MERGE

Comandos transaccionales

Por defecto, MySQL se ejecuta con el modo autocommit activado. Esto significa que en cuanto ejecutemos un comando que actualice (modifique) una tabla, MySQL almacena la actualización en disco.

Si usamos tablas transaccionales (como InnoDB), podemos desactivar el modo autocommit con el siguiente comando: SET AUTOCOMMIT=0;

Tras deshabilitar el modo autocommit poniendo la variable AUTOCOMMIT a cero, debemos usar **COMMIT** para almacenar los cambios en disco o **ROLLBACK** si se quiere ignorar los cambios hechos desde el comienzo de la transacción.

Si se quiere deshabilitar el modo autocommit para una serie única de comandos, podemos usar el comando **START TRANSACTION**:

START TRANSACTION;
SELECT @A:=SUM(salario) FROM tabla1 WHERE tipo=1;
UPDATE tabla2 SET suma=@A WHERE tipo=1;
COMMIT;

Con START TRANSACTION, autocommit permanece deshabilitado hasta el final de la transacción con COMMIT o ROLLBACK. El modo autocommit vuelve a su estado previo.

BEGIN y **BEGIN WORK** se soportan como alias de START TRANSACTION para iniciar una transacción.

Procedimientos almacenados

Un procedimiento almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales pero pueden en su lugar referirse al procedimiento almacenado.

Algunas situaciones en que los procedimientos almacenados pueden ser particularmente útiles son:

- Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la base de datos.

- Cuando la seguridad es muy importante. Los bancos, por ejemplo, usan procedimientos almacenados para todas las operaciones comunes. Esto proporciona un entorno seguro y consistente y los procedimientos pueden asegurar que cada operación se logra apropiadamente.

Los procedimientos almacenados pueden mejorar el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente.

Sintaxis: Los procedimientos almacenados y rutinas se crean con comandos CREATE PROCEDURE y CREATE FUNCTION. Una rutina es un procedimiento o una función. Un procedimiento se invoca usando un comando **CALL** y sólo puede pasar valores usando variables de salida. Una función puede llamarse desde dentro de un comando como cualquier otra función (esto es, invocando el nombre de la función) y puede retornar un valor escalar. Las rutinas almacenadas pueden llamar otras rutinas almacenadas.

CREATE PROCEDURE *sp_nombre* ([parámetro[,...]])
[características...] cuerpo

Disparadores (triggers)

A partir de MySQL 5.0.2 se incorporó el soporte básico para triggers. Un disparador es un objeto con nombre dentro de una base de datos el cual se asocia con una tabla y se activa cuando ocurre en ésta un evento en particular. Sintaxis:

CREATE TRIGGER *nombre_disp momento_disp evento_disp*
ON *nombre_tabla* FOR EACH ROW *sentencia_disp*

Programas

MySQL AB también suministra tres programas con interfaz gráfica de usuario para utilizar con el servidor de bases de datos MySQL:

- **MySQL Administrator**. Esta herramienta se emplea para la administración de servidores, bases de datos, tablas y usuarios de MySQL (figura 2.28).

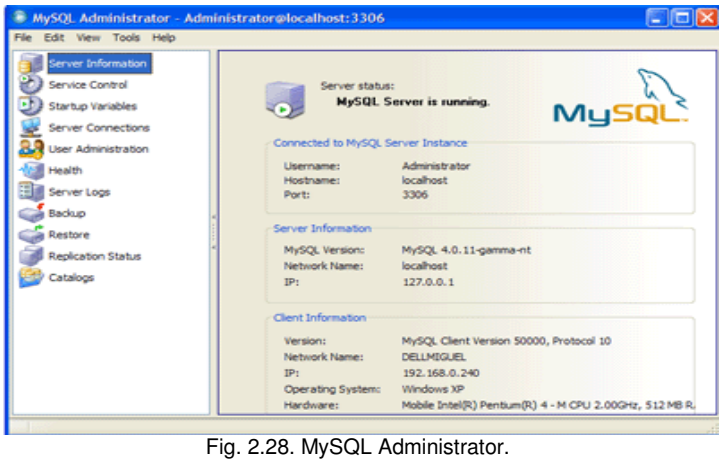


Fig. 2.28. MySQL Administrator.

- **MySQL Query Browser**. Esta herramienta gráfica es provista por MySQL AB para crear, ejecutar y optimizar consultas dirigidas a bases de datos MySQL.

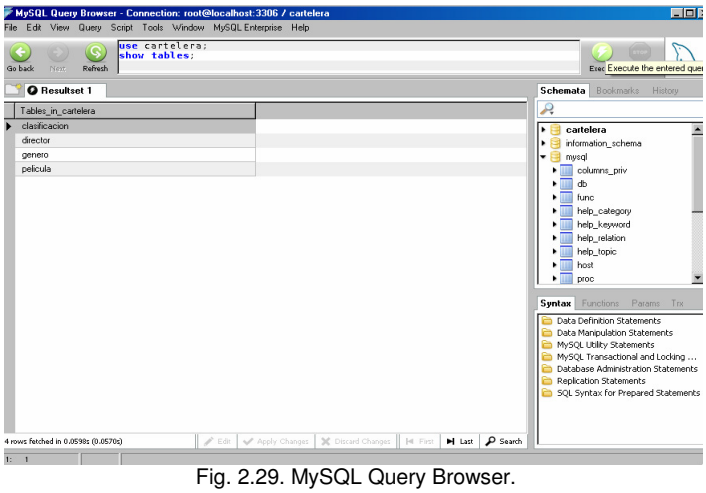


Fig. 2.29. MySQL Query Browser.

MySQL Migration Toolkit. Herramienta orientada a brindar asistencia en el proceso de migración de esquemas y datos desde otros sistemas gestores de bases de datos relacionales hacia MySQL.

2.7 Módulo VII. Introducción a la seguridad en cómputo

Conceptos básicos

Seguridad. Es la característica de un sistema o elemento que permite garantizar que se opera como se espera que lo haga, que es ajeno a todo riesgo y a toda amenaza, que no posee vulnerabilidades, que es confiable y que funciona sin fallo.

Por otro lado, una **amenaza** es una circunstancia o evento (robo, infección por virus de cómputo, etc.) que puede causar daño a un sistema, frecuentemente aprovecha una vulnerabilidad.

Vulnerabilidad. Se refiere a la ausencia de una contramedida, o debilidad de la misma. Predisposición de un sistema a ser afectado por un agente perturbador, la debilidad puede originarse en el diseño, la implementación o en los procedimientos para operar y administrar el sistema. En seguridad informática se denomina “hoyo”.

Peligro. Es la probabilidad de que se presente una amenaza.

Grado de exposición. Nivel de afectación, número de personas, bienes o módulos de sistema que podrían ser afectados.

Riesgo. Es la probabilidad de que una vulnerabilidad sea explotada, de acuerdo a su nivel de exposición y al peligro involucrado.

Riesgo = Peligro * Exposición * Vulnerabilidad

Entonces, un sistema de cómputo es seguro si se puede confiar en que se comportará como se espera que lo haga, que la información en él se mantendrá inalterada y accesible durante el tiempo que su dueño lo desee para los usuarios que él mismo determine.

Servicios y políticas de seguridad

Un servicio de seguridad es una característica que debe tener un sistema para satisfacer una política de seguridad.

Una política de seguridad especifica las características de seguridad que una organización debe observar y proveer con el fin de salvaguardar su información.

Estándares como el BS 7799 o el ISO 17799 consideran a la información como un activo y definen servicios para su protección.

Confidencialidad. Sólo el propietario del secreto es capaz de descifrar la información.

Autenticación. Se asegura de la identidad de la persona del otro lado de la línea. Se puede realizar mediante alguno de los siguientes mecanismos:

- Basados en algo que se sabe. Primeros sistemas de autenticación, se basan en claves de acceso: nombre usuario, claves de acceso, nips, passwords, etc. Son fáciles de usar y no requieren de un hardware especial. Siguen siendo el sistema de autenticación más usado hoy en día.

Otros mecanismos de autenticación son los mostrados en las figuras 2.30 y 2.31.

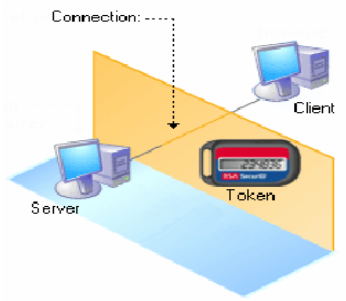
- 

Fig. 2.30. Basados en lo que se tiene.

- 

Fig. 2.31. Basados en lo que se "es".

Integridad. Se asegura que la información se mantenga inalterada desde su creación.

Autorización (Control de Acceso). Se asegura que la información estará disponible sólo para determinados usuarios, con la posibilidad de manejar diferentes niveles de acceso para cada uno de ellos.

No repudio. Se asegura que el emisor de la información no puede negar haberla enviado.

Disponibilidad. Asegurar que el sistema este disponible cuando se le requiera.

Auditoria. Asegurar que se defina un registro cronológico de los eventos exitosos y fallidos, que proporcionen evidencia de la actividad del sistema.

La criptología como herramienta en seguridad informática

La **criptografía,** del griego "Kryptos" oculto y "graphein" escritura (escritura oculta), se encarga de convertir un texto normal y comprensible en un formato incomprensible a menos que se posea un conocimiento secreto. En épocas recientes la criptología se define como la ciencia de usar las matemáticas para cifrar y descifrar información.

El **criptoanálisis** es la ciencia de analizar y romper la comunicación segura mediante el análisis del algoritmo empleado.

Los algoritmos criptográficos pueden estar basados en:

- **Criptografía simétrica**. Requiere que el emisor y el receptor compartan una clave secreta "llave", la cual es utilizada para cifrar el mensaje cuando se envía y descifrar el mensaje al recibirlo (figura 2.32). El gran inconveniente se presenta en el proceso de intercambiar de forma segura la llave.

En este esquema de seguridad es importante que el emisor disponga de un canal seguro para realizar la entrega de la llave al inicio de la transmisión. Ejemplos: DES, 3DES, AES, Blowfish, IDEA.

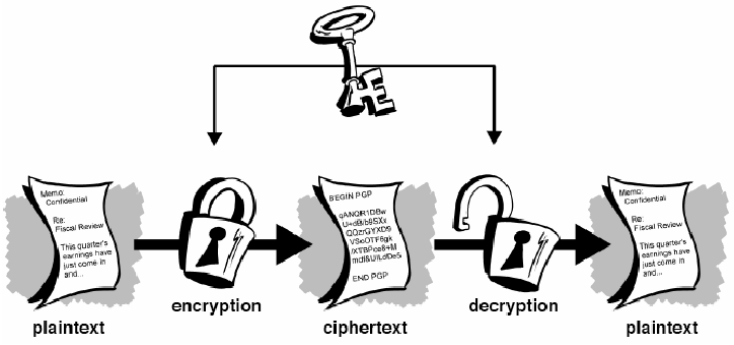


Fig. 2.32. Criptografía simétrica.

- **Criptografía Asimétrica** (también conocida como de llave publica). Resuelve el problema de intercambiar el secreto utilizando para ello un algoritmo basado en dos llaves, tanto el emisor como el receptor utilizan un par de llaves, una "pública" y otra "privada". La privada es mantenida en secreto por un individuo, la pública esta disponible para que otro individuo sea capaz de cifrar la información sensible con ella (figura 2.33).

  Cuando el emisor desea enviar un mensaje lo cifra con la llave pública del receptor, siendo éste el único capaz de descifrarlo mediante el uso de su llave privada. En este caso conseguimos asegurar la confidencialidad del mensaje. Rivest Shamir Adleman (RSA) y Diffie-Hellman (D-H) son dos sistemas de llaves públicas utilizados actualmente.

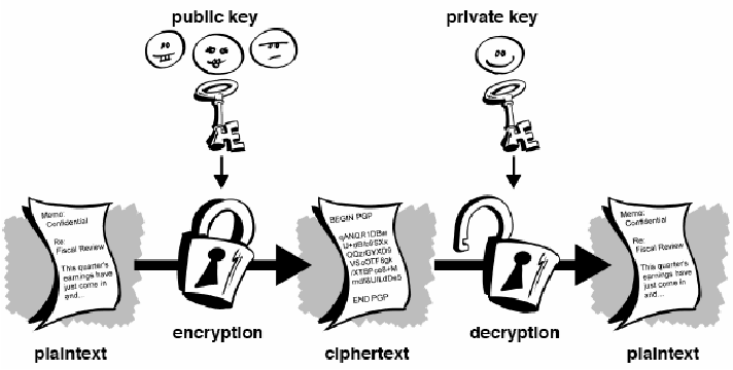


Fig. 2.33. Criptografía asimétrica.

Message digest. Conocidos también como Hashes, son el resultado de algoritmos unidireccionales que permiten generar una "huella digital" de un mensaje. Permiten obtener una cadena de longitud fija que es una representación condensada de un mensaje. Permite garantizar la integridad de un mensaje, si un sólo carácter del mensaje original cambia el Hash será totalmente distinto. Ejemplos: SHA1, SHA2, MD5.

Gnu Privacy Guard (GnuPG). Es un sistema de codificación de código libre y desarrollo abierto. Es un reemplazo libre a PGP (Pretty Good Privacy).

Instalando GPG

- http://www.gnupg.org/(en)/download/index.html
- ./configure --prefix=/usr/remote/gpg
- make
- make install

Usando GPG

gpg --gen-key	Se genera un nuevo par de llaves (privada y publica) para el usuario.
gpg --list-keys	Muestra las llaves con las que cuenta el "llavero".
gpg --export --armor > MiLlave.key	Se exporta la llave pública del usuario a un archivo.
gpg --import < LlaveDeUnAmigo.key	Se importa la llave pública de otro usuario.
gpg --sign-key <id>	Se firma la llave pública de otro usuario y se considera confiable.
gpg -e Mensaje.txt	Se cifra un Mensaje.txt con la llave de un usuario.
gpg -d Mensaje.txt.gpg	Se descifra el Mensaje.txt con la llave privada del usuario.
gpg -s Mensaje.txt	
gpg --sign Mensaje.txt	Se firma un Mensaje.txt con la llave de un usuario.
gpg --clearsign Mensaje.txt.gpg	Se firma el Mensaje.txt en texto claro con la llave de un usuario.
gpg -v Mensaje.txt.gpg	
gpg --verify Mensaje.txt.gpg	Se verifica la firma del Mensaje.txt.

Open Secure Socket Layer (OpenSSL). Es un desarrollo de código abierto con el propósito de desarrollar un Toolkit de SSL y TLS con el nivel de una herramienta profesional. Está basado en el código fuente de SSLeay.

Instalar openssl

- http://www.openssl.org/source/openssl-0.9.7e.tar.gz
- ./config --prefix=/usr/remote/openssl
- make
- make install

Cifrando mensajes con openssl → openssl enc -e -des3 -in Ejemplo.txt -out Ejemplo.des3
Descifrando mensajes con openssl → openssl enc -d -des3 -in Ejemplo.des3 -out Ejemplo.txt

Message Digest

openssl dgst -md5 Ejemplo.txt → MD5(Ejemplo.txt)= 08a4ebfb7bf8f68a4e169a9806cb3a3b
md5sum Ejemplo.txt → 08a4ebfb7bf8f68a4e169a9806cb3a3b Ejemplo.txt
openssl dgst -sha Ejemplo.txt → SHA(Ejemplo.txt)=37ad34ca6b2d8447a7504b3c76aa7d8bb319644f

Aplicando seguridad en el servidor Web Apache

Existen múltiples recomendaciones que se pueden realizar para mejorar la seguridad del servidor de Web, entre las más importantes tenemos:

- Mantener en producción una versión actualizada.
- Evitar el uso de permisos no necesarios en los directorios de Apache.
- Restringir el uso de SSI y de CGIs.
- Revisar con frecuencia las bitácoras del Web server.
- Seleccionar passwords fuertes para los accesos restringidos.
- Implementar el soporte para SSL (mod_ssl).

- Agregar módulos orientados a incrementar el nivel de seguridad (mod_access, mod_auth, mod_security).

Considerando que los medios de transmisión no son confiables, es necesario establecer mecanismos para evitar que la información sea capturada durante su transmisión, para lo cual se utiliza: SSH, POPS, IMAPS, SFTP, etc.

En el caso de HTTP implementamos el uso de HTTPS que utiliza SSL para cifrar su transmisión. Diseñado por Netscape en 1993, SSL es una propuesta de estándar para cifrado y autenticación en el Web.

Es un esquema de cifrado de bajo nivel usado para transacciones en protocolos de nivel aplicación como HTTP, FTP, etc. Con SSL puede autentificarse un servidor con respecto a su cliente y viceversa.

Objetivos de SSL

- **Seguridad criptográfica**. Se sugiere el uso de SSL para establecer conexiones seguras entre dos partes.
- **Interoperabilidad**. Programadores deben poder desarrollar aplicaciones basadas en SSL que intercambien parámetros criptográficos sin tener conocimiento de los códigos de los programas de cada uno.
- **Extensibilidad**. Provee un marco donde pueden incorporarse métodos criptográficos según se necesite.

Certificados digitales. La norma X.509 es el estándar para formatos de certificados con llave pública. Un certificado X.509 consiste de la llave pública de un usuario y la firma de un tercero para la identificación de ese usuario.

Autoridades certificadoras (CA). El esquema de firmas digitales requiere de alguien que autentique que un individuo es quien dice ser. Instituciones gubernamentales, comerciales o financieras se encargan de emitir certificados digitales. En los certificados se integra la llave pública del individuo, piezas de información sobre el individuo e identificadores de la autoridad certificadora quien finalmente se encargará de firmar digitalmente el certificado con su llave privada.

HTTPS

1. El cliente establece una conexión a un servidor que soporta HTTPS.
2. Se negocian los parámetros de comunicación. El cliente notifica al servidor cuáles son los parámetros que soporta y éste selecciona cuales de ellos utilizará. Entre los parámetros tenemos: qué algoritmos de cifrado se utilizarán, la versión del protocolo, etc.
3. El servidor envía al cliente su certificado digital (x509).
4. El cliente utiliza su copia de la llave pública de la CA, la fecha y el nombre del servidor para validar el certificado.
5. Se realiza un acuerdo de llave, el cliente genera un valor aleatorio “clave premaestra”, lo cifra con la llave pública del servidor y se lo envía.
6. El servidor debe descifrar con su llave privada la “clave premaestra” (autenticación).
7. Tanto el cliente como el servidor deben generar la “clave maestra” a partir de la “premaestra”.
8. El cliente y el servidor intercambian cifrado el Message Digest de la “clave maestra” para asegurar que obtuvieron lo mismo. (integridad y confidencialidad).
9. A partir de la “clave maestra” ambos generan la clave de sesión, con la cual se cifrará la comunicación.

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

Ataques pasivos

- Recopilar información en Bases de Datos.
- Whois (ARIN,LACNIC).
- DNS – NIC (listados de IPs y nombres de dominio, trasferencias de zonas).
- Trazado de ruta(traceroute).
- Sniffers (snort, dsniff).
- Analizadores de tráfico (ethereal).

Sniffers. Es un proceso que "olfatea" el tráfico que se genera en la red, es capaz de leer toda la información (posiblemente claves de acceso) que circule en el segmento de red en el que se ubique (figura 2.34). Un analizador de protocolos es un Sniffer al que se le ha añadido funcionalidad suficiente como para entender y traducir los protocolos que se están hablando en la red.

Fig. 2.34. Un Sniffer.

Tipos de Sniffers

- Pasivos (no realizan actividad alguna sólo capturan paquetes).
- Activos (intentan apoderarse de las sesiones).
- Spoofing.
- Envenenamiento de la tabla de ARP[5].

Son difíciles de detectar y combatir ya que son programas pasivos y no generan bitácoras. Cuando se usan propiamente utilizan un mínimo de CPU y memoria. Es posible localizarlos a nivel local y, a nivel red, algunos pueden localizarse mediante herramientas como antisnif.

Ataques activos

- **Escaneo de Puertos**. Es un método que consiste en conectarse a los puertos UDP y TCP del sistema objetivo con el propósito de determinar cuáles se encuentran escuchando. **Nmap** es un programa que realiza un escaneo para verificar qué puertos se encuentran disponibles en el servidor objetivo.

[5] **ARP** son las siglas en inglés de **A**ddress **R**esolution **P**rotocol (protocolo de resolución de direcciones).

- **Ataque de Fuerza Bruta**. Se busca probar todas las posibles opciones, se recorre por completo el universo con la intención de obtener acceso al sistema o descifrar cierta información.

- **Ataque por diccionario**. Sólo se prueban aquellas opciones con mayor probabilidad de éxito las cuales se encuentran en un diccionario.

Hydra. Es una herramienta para realizar ataques por fuerza bruta sobre una gran variedad de servicios de red, entre los que destacan: telnet, ftp, pop3. Imap, smb, http-get, rlogin, vnc, ssh2, ldap,mysql, etc.

John The Ripper. Tradicionalmente el primer script para ataques por diccionario, su principal objetivo era el obtener claves débiles del archivo de password.

Secuestro de sesiones. Tipo de ataque en el que el atacante toma control de una comunicación tal y como un secuestrador de aviones toma control del avión. El objetivo es robar una conexión generada por una aplicación de red iniciada por un cliente.

Denial of Service (DOS). Todo ataque que busca aparentar una alta demanda sobre un servicio valido con el objeto de saturar la capacidad de respuesta y generar una caída del sistema victima (Ping de la muerte, Inundación Syn, Spoofing, Smurf, Fraggle, DOS distribuido).

- **Ping of Death**. Se lanza un paquete que excede el tamaño máximo (65535 bytes de datos) permitidos por la especificación del protocolo IP a un equipo victima, si el equipo es susceptible a este ataque puede sufrir un “crash” reinicio.

- **Smurfing**. Es uno de los ataques de DOS más temidos, requiere 3 actores: La víctima, el atacante y la red amplificadora. El atacante originará un paquete hacia la dirección de broadcast[6] de la red amplificadora, haciendo aparecer que su origen es una interfaz de la red de la víctima. Cada interfaz de la red amplificadora enviará respuestas a la supuesta interfaz de origen creando un tráfico que la victima no es capaz de soportar. Para no permitir ser utilizado como red amplificadora debemos deshabilitar el paso de mensajes destinados a broadcast a través de los routers[7] de frontera.

- **Spoofing**. Es la creación de paquetes de comunicación TCP/IP usando una dirección IP de alguien más. Lo anterior permite entrar en un sistema haciéndose pasar por un usuario autorizado. Una vez dentro del sistema, el atacante puede utilizarlo como plataforma para introducirse en otro y así sucesivamente.

Ingeniería social. Es una de las formas más comunes para penetrar sistemas de “alta seguridad”. Uso de trucos psicológicos por parte de un atacante externo sobre usuarios legítimos de un sistema para obtener información (usenames y passwords) necesaria para acceder a un sistema. Se basa en ataques como usurpación de identidad, hurgar en la basura, inocencia de la gente, relaciones humanas, etc.

Virus. Se define como una porción de código de programación cuyo objetivo es implementarse a si mismo en un archivo ejecutable y multiplicarse sistemáticamente de un archivo a otro. Además de esta función primaria de "invasión" o "reproducción", los virus están diseñados para realizar una acción concreta en los sistemas informáticos sin la autorización del usuario.

---

[6] **Broadcast** es un modo de transmisión de información donde un nodo emisor envía información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.

[7] Dispositivo de hardware para interconexión de redes de computadoras que opera en la capa tres (nivel de red) del modelo OSI (Open System Interconnection).

Los gusanos. Es un programa que produce copias de sí mismo de un sistema a otro a través de la red. En las máquinas que se instala produce enormes sobrecargas de procesamiento que reducen la disponibilidad de los sistemas afectados.

Fig. 2.35. Gusanos en el sistema.

Bombas lógicas. Es una modificación en un programa que lo obliga a ejecutarse de manera diferente bajo ciertas circunstancias. En condiciones normales el programa se comporta como previsto y la bomba no puede ser detectada.

"La diferencia es que en los ataques pasivos sólo se observa o se 'olfatea' a la victima y en los activos se ataca con un propósito específico".

Exploit. Se refiere a la forma de explotar una vulnerabilidad.

Firewall. Existen dos tipos básicos:

- **Hardware Firewall**: Usualmente es un ruteador o switch capa 3, dispone ciertas reglas para dejar pasar o no los paquetes.
- **Software Firewall**: Es un programa que verifica los paquetes con diferentes criterios para dejarlos pasar o descartarlos.

Firewall en Linux. iptables/Netfilter son las dos piezas principales de producto Firewall disponibles gratuitamente para distribuciones Linux, iptables es usado para construir las reglas, Netfilter es puente entre el núcleo de Linux y las iptables.

Entre los principales ataques sobre sistemas Web tenemos:

SQL inyection. Es un ataque que aprovecha la mala validación de las entradas de usuario para inyectar instrucciones SQL que no corresponden a las realizadas de forma normal por el sistema. Permite el acceso a la base de datos, se atenta contra la integridad y confidencialidad de la información que reside en ella.

Cross Site Scripting – XSS. Busca presentar un contenido distinto al original a un visitante específico de un sitio. No modifica realmente el sitio, solo lo muestra de forma distinta. Permite mostrar información falsa en un sitio autentico, se atenta contra la integridad de la información que publica el sitio.

Herramientas de seguridad para sistemas Web

Nikto es un analizador de vulnerabilidades conocidas para sitios Web.

http://www.cirt.net/code/nikto.shtml

Ejemplo de uso: ./nikto.pl -h www.victima.com.mx -p 80 -v

2.8 Módulo VIII. Desarrollo de aplicaciones con PostgreSQL y PHP

Programación orientada a objetos (POO)

Es una metodología de programación que se fundamenta en el concepto de objeto. Se trata de una filosofía donde la realidad se modela mediante objetos y la interacción entre ellos.

Ventajas:
- La forma de programar pasa a un segundo plano, permite agregar funcionalidad sin incrementar en la misma proporción la complejidad.
- Los programas no son sólo líneas que se ejecutan una tras otra.
- Permite identificar entidades que conviven en un sistema dado.
- La herencia permite la reutilización de código
- Favorece la generación de bancos de componentes.

Ejemplos de lenguajes de programación que proporcionan soporte para la programación orientada a objetos son: Java, SmallTalk, Delphi, Perl, C++, C#, y evidentemente PHP.

Características de la POO

Abstracción
- Surge de reconocer las similitudes entre ciertos objetos, situaciones o procesos del mundo real.
- Se enfatiza en detalles significativos.
- Se suprime lo irrelevante.

Encapsulamiento
- Consiste en almacenar en un mismo "lugar" los elementos de una abstracción que constituyen su estructura y funcionamiento.

Modularidad
- Se refiere a fragmentar un programa en componentes para reducir su complejidad.
- Repercute en el acoplamiento o integración de los componentes.

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

Clase
- Una clase es una colección de datos y métodos que operan sobre ellos.
- Es la definición de un nuevo tipo de dato (figura 2.36).
- Es el "template" ó "plantilla" mediante la cual se crean los objetos.

Circulo.php

```
1: <?php
2:      class Circulo
3:      {
4:          /*
5:          Definición de la Clase.
…
10:         */
11:     }
…
15: ?>
```

Fig. 2.36. Una clase en PHP.

Objeto

- Es la representación lógica de un elemento físico.
- Es la implementación de una clase.

Busca representar un objeto físico o lógico de la realidad, como puede ser una silla, un automóvil, un circulo, etc. (figura 2.37).

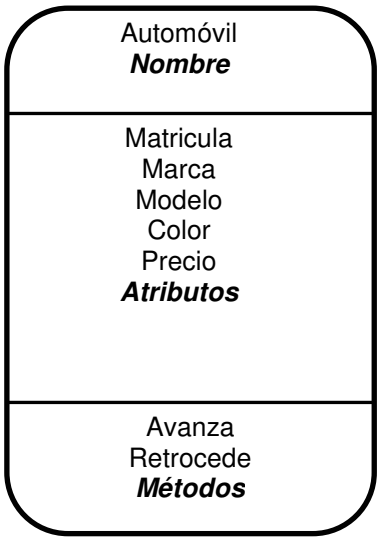


Fig. 2.37. Una clase en PHP.

Los **atributos** describen las características de los objetos. Poseen un tipo, que nos indica qué clase de atributo es. Si bien existen ciertos tipos primitivos, como enteros, boléanos y reales, cualquier tipo puede ser usado, incluso otras clases.

Un **método** es el procedimiento o función que se invoca para actuar sobre un objeto y sirve para que éste pueda ejecutar sus responsabilidades (figura 2.38). Los métodos determinan como actúan los objetos cuando reciben un mensaje. Un mensaje es el resultado de cierta acción efectuada por un objeto. Al conjunto de mensajes a los cuales puede responder un objeto se le conoce como protocolo del objeto.

```
<? php
class Automovil
{
  var $marca="VW";
  var $modelo="Sedan";
  var $color="Negro";
  function avanza () { … }
  function retrocede () { … }
  function estado ()
  {
    print "Descripcion del Automovil ";
    print "Color : " . $this->color . " <BR>";
    print "Modelo: " . $this->modelo . " <BR>";
    print "Marca : " . $this->marca . " <BR>";
  }
}
$auto=new Automovil();
$auto->avanza();
$auto->retrocede();
$auto->estado();
?>
```

Fig. 2.38. Métodos y atributos.

El método principal de un objeto se denomina constructor. Sirven para:

- Inicializar los atributos del objeto.
- Reservar la memoria necesaria para mantener el objeto.

Constructores en PHP

Previo a PHP 5, requerían tener el mismo nombre que la clase que los contenía, ahora se deben llamar **__construct()**, doble guión bajo al inicio del nombre.

Firma de un método. Se denomina así al encabezado de cualquier método. Se compone básicamente de los siguientes elementos: tipo de acceso, nombre y parámetros.

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

Herencia

- Las nuevas clases se construyen a partir de las ya existentes (figura 2.39).
- Permite el paso de variables y métodos.
- Reutilización de código.
- Permite la especialización de objetos.

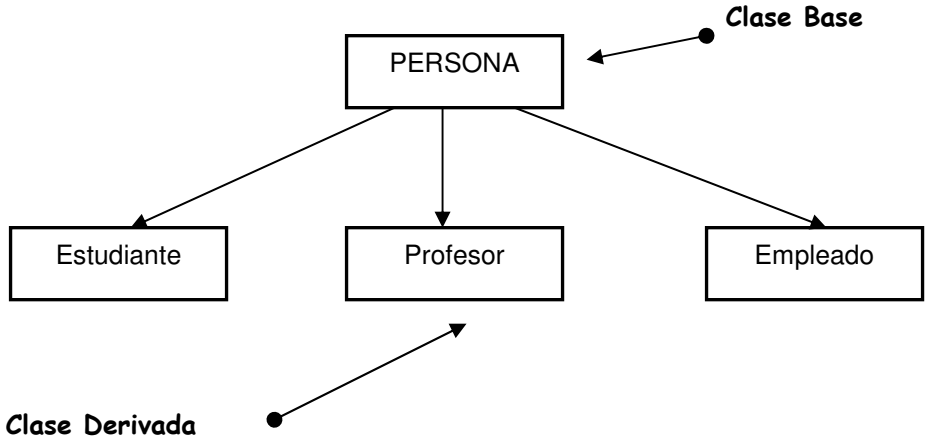


Fig. 2.39. Herencia.

Existen básicamente dos tipos de herencia: la simple y la múltiple. La herencia en PHP es simple, no se soporta la herencia múltiple.

- Se utiliza la palabra reservada **extends** en el encabezado de la clase.
- Se lee 'hereda de'.
- Se heredan métodos, variables y constructores.

Visibilidad. Es el nivel de protección para acceder a un atributo o método de una clase.

- **Public**: Pueden ser accesados desde cualquier parte.

- **Protected**: El acceso está limitado a las clases hijas así como a la clase que define el objeto.
- **Private**: El acceso está limitado sólo a la clase que define el objeto.

Interfaces. Tipos específicos de clases, buscan asegurar que la clase que las implementen integre como parte de su definición determinadas características.

- Son muy parecidas a la definición de una clase.
- Todos los métodos se declaran sin cuerpo, únicamente va la firma.
- Las clases que la utilizan deben usar la palabra reservada "**implements**".
- Las clases deben tener un nombre distinto al de las interfaces que implementen.

Excepciones. PHP 5 integra soporte para el manejo de excepciones, la intención es tomar en cuenta situaciones anormales dentro de la ejecución del programa y actuar en consecuencia.

- Son muy parecidas a la definición de una excepción en Java.
- Se utilizan las instrucciones try / catch para capturar las excepciones.
- Se utiliza la instrucción throw para lanzar una nueva excepción.
- PHP no lanza las excepciones es necesario lanzarlas manualmente.

Manejador de bases de datos PostgreSQL

Es un manejador de bases de datos objeto-relacional altamente escalable, compatible con SQL y con más de 15 años de desarrollo. Surge en la Universidad de Berkeley y está basado en Ingres.

PostgreSQL es un descendiente del código abierto y ofrece muchas características modernas:

- Consultas complejas.
- Llaves foráneas.
- Procedimientos almacenados.
- Disparadores.
- Vistas.
- Control de concurrencia transaccional.

Instalación y configuración de PostgreSQL

Descargar. Podemos descargar PostgreSQL desde la sección de descargas de su sitio Web (http://www.postgresql.org) el cual tiene varios mirrors. Para la mayoría de los usuarios de PostgreSQL que tienen sistemas tipo Unix, se recomienda que descarguen y compilen el código fuente.

Agregar Grupo. Antes de crear el usuario que será el administrador del servidor de PostgreSQL, es recomendable generar un grupo y, a partir de éste, agregar a los usuarios de PostgreSQL. Para ello es necesario utilizar el comando:

$addgroup *postgresql*

Agregar Usuario. El siguiente paso es crear un usuario al que agregaremos al grupo creado en el paso anterior:

$useradd *–g postgresql* postgresql

Descomprimir. Extraer el código fuente del archivo .tar.gz que acabamos de descargar:

$tar –zxvf postgresql-xx.tar.gz

Estos comandos crearán un nuevo directorio dentro del que actualmente nos encontramos y que contendrá el código fuente de la distribución. Debemos cambiarnos a ese directorio con **cd** para proceder a compilar el servidor PostgreSQL.

Configuración. Sólo tenemos que ejecutar el siguiente comando:

$./configure –prefix = /usr/local/postgresql

Como podemos observar el prefix es la ruta en la que se instalará el servidor de PostgreSQL.

Compilar. Ahora podemos compilar las diferentes partes que forman PostgreSQL simplemente ejecutando el siguiente comando:

$make

Instalar. Es el momento de instalar el paquete en el directorio elegido en prefix:

$make install

Para comprobar que nuestro servidor funciona correctamente hay que seguir los siguientes pasos:

- El primer paso es ubicarnos en el directorio donde se instaló el servidor:

$cd /usr/local/postgresql/bin

- El siguiente paso es generar un directorio donde se crearan los archivos necesarios para levantar el servidor y generar una base de datos.

$./initdb directorio

- Por último, levantamos el servidor, para ello utilizamos el comando:

$./pg_ctl start –D directorio

Para detener el servidor:

$./pg_ctl stop –D directorio

Cuando iniciamos una sesión es PostgreSQL podemos ver una pantalla semejante a la que nos muestra la figura 2.40.

```
$ su - postgres
$ createdb prueba

Welcome to the psql, the PostgreSQL interactive
terminal.
Type \copyright for distribution terms
\h for help with SQL commands
\? For help on internal slash commands
\g or terminate with semicolon to execute query
\q to quit
prueba=#
```

Fig. 2.40. Una sesión en PostgreSQL.

Los tipos de datos que soporta PostgreSQL, la sintaxis y la información general se pueden consultar en la documentación.

Más características de PostgreSQL

- **Atomicidad (indivisible)**. Es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- **Consistencia**. Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- **Aislamiento**. Es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
- **Durabilidad**. Es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.
- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc.
- Soporte de todas las características de una base de datos profesional (triggers, store procedures–funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.).
- Soporte de tipos de datos de SQL92 y SQL99.
- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.
- Utilidades para limpieza de la base de datos (Vacuum).
- Utilidades para análisis y optimización de querys.

Además, cuenta con programas de diseño y administración como lo muestran las figuras 2.41 y 2.42.

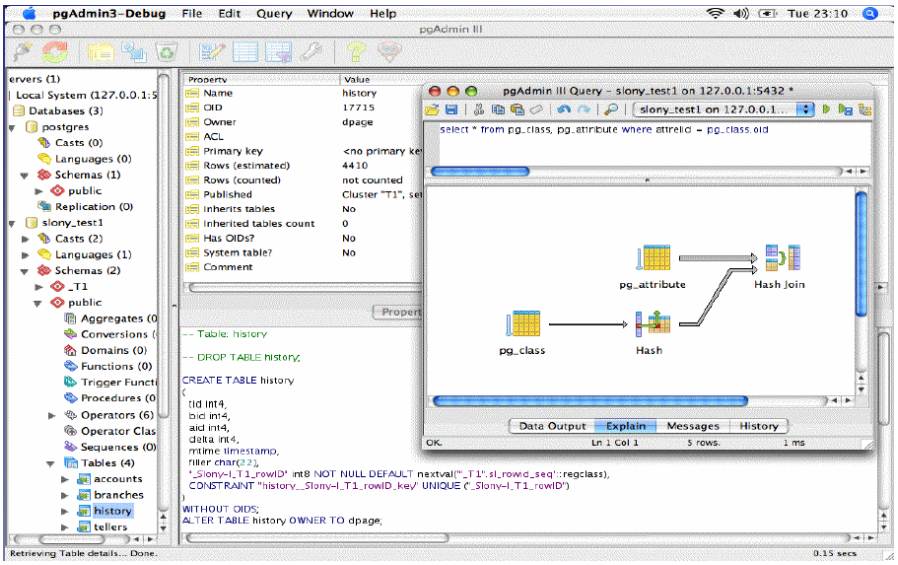


Fig. 2.41. PgAdmin.

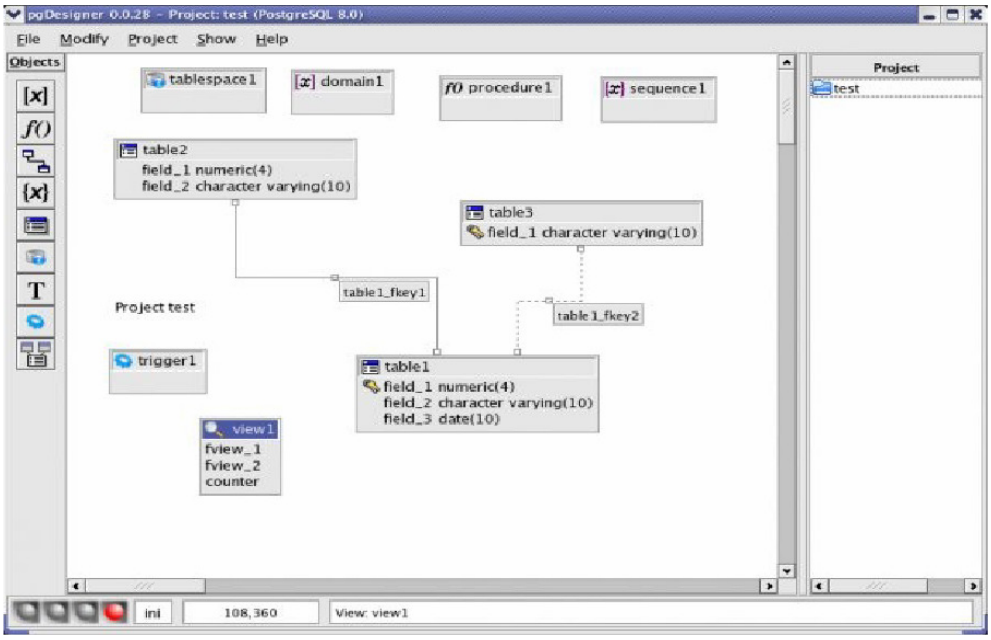


Fig. 2.42. PgDesigner.

2.9 Otras Herramientas

ADODB

Como hemos visto, PHP está especialmente diseñado para la creación de sitios Web dinámicos y, para crear estos sitios normalmente se utiliza algún tipo de base de datos desde donde obtenemos la información que queremos mostrar, ya sean noticias, preguntas y respuestas de un foro u otro tipo de información dinámica. El problema aparece cuando nuestro proyecto crece tanto que necesita hacer uso de otro tipo de base de datos más robusta.

Desafortunadamente el acceso en PHP a cada base de datos es muy diferente. Para conectarnos a MySQL, debemos usar mysql_connect(); cuando utilizamos PostgreSQL, usamos pg_connect(). Lo peor es que también los parámetros de cada función son diferentes.

Una librería de abstracción de los datos como ADODB es lo que se necesita si deseamos asegurar en gran medida la portabilidad de una aplicación de este tipo. Provee una serie de funciones comunes para comunicarse con las distintas bases de datos.

ADODB viene de "Active Data Objects DataBase". Actualmente soporta MySQL, PostgreSQL, Oracle, Interbase, Microsoft SQL Server, Access, FoxPro, Sybase, ODBC y ADO. Podemos ver un ejemplo de aplicación en las figuras 2.43 y 2.44.

```
$db = mysql_connect("localhost", "root", "password");
mysql_select_db("mydb",$db);
$result = mysql_query("SELECT * FROM empleados",$db);
if ($result === false) die("failed");
while ($fields = mysql_fetch_row($result))
{
    for ($i=0, $max=sizeof($fields); $i < $max; $i++)
    {
        print $fields[$i].' ';
    }
    print "<br>n";
}
```

Fig. 2.43. Ejemplo con MySQL.

```
include("adodb.inc.php");
$db = NewADOConnection('mysql');
$db->Connect("localhost", "root", "password", "mydb");
$result = $db->Execute("SELECT * FROM empleados");
if ($result === false) die("failed");
while (!$result->EOF)
{
    for ($i=0, $max=$result->FieldCount(); $i < $max; $i++)
        print $result->fields[$i].' ';
    $result->MoveNext();
    print "<br>n";
}
```

Fig. 2.44. Su equivalente en ADODB.

Ahora portarlo a PostgreSQL es tan simple como cambiar la segunda línea a:

NewADOConnection('postgres')

Conectando a la base de datos

```
include("adodb.inc.php");
$db = NewADOConnection('mysql');
$db->Connect("localhost", "root", "password", "mydb");
```

La conexión puede parecer algo más complicada que en MySQL o PostgreSQL pero se explica en que ADODB es totalmente orientado a objetos.

Para ahorrar memoria sólo se cargan las funciones específicas de la base de datos que vamos a utilizar. Cada driver tiene un archivo distinto que se carga mediante un **include** automáticamente. Una vez cargado el driver que vamos a utilizar mediante la función NewADOConnection() nos conectamos a la base de datos usando $db->Connect().

Ejecutando SQL

```
$result = $db->Execute("SELECT * FROM empleados");
if ($result === false) die("failed");
```

Para enviar una sentencia SQL al motor de base de datos se utiliza la función Execute(). Ésta devuelve un objeto "recordset" si la ejecución fue correcta o un "false" si hubo algún error.

El objeto que realiza la conexión a la base de datos ejecuta las sentencias SQL y tiene otro “set” de funciones para estandarizar el formato de sentencias SQL como la concatenación de cadenas o formatos de fechas.

Obteniendo datos

```
while (!$result->EOF)
{
    for ($i=0, $max=$result->FieldCount(); $i < $max; $i++)
        print $result->fields[$i].' ';
    $result->MoveNext();
    print "<br>n";
}
```

El proceso para obtener datos es similar al que se efectúa para leer desde un fichero. Para cada línea observamos si hemos llegado al final del fichero (EOF). Mientras no lleguemos al final seguimos leyendo y moviéndonos a la siguiente línea (MoveNext).

ADODB es lo suficientemente robusto y completo para casi cualquier aplicación. Está siendo actualizado permanentemente y tiene muchos seguidores por lo que es una muy buena opción.

JavaScript

Es un lenguaje de programación que originalmente fue creado por la empresa Netscape para añadir interactividad a las páginas Web vistas con su navegador de Internet, actualmente está soportado por casi cualquier navegador gráfico que podamos encontrar, es interpretado y tiene una sintaxis semejante a la del lenguaje Java.

El código de JavaScript se embebe en el código HTML de las páginas Web añadiendo cierta "inteligencia" e interactividad a las mismas. Su uso bien es para obtener ciertos efectos estéticos (cambiar una imagen al pasarle por encima, mover un gráfico por la pantalla, etc.), para validar una entrada de datos, hacer cálculos, etc.

En las figuras 2.45 y 2.46 se muestra un ejemplo de validación de un formulario, no permitiendo campos vacíos:

```
<html>
<head>
   <title>Valida entrada de usuario</title>
   <script language="javascript">
        <!--
        function validaCampo(objeto){
            //comprueba si el texto del objeto tiene longitud cero
            if(!objeto.value.length)
                alert("Este campo no puede estar vacio.");
        }
        -->
   </script>
</head>
<body>
  <form method="post" action="ejemplo.html" onsubmit=" if
    (this.nombre.value.length==0){
        alert('El campo \"Nombre\" debe tener contenido');
        return false;
    return true;">

    Nombre:<input  type="text" name="nombre" size="20">
    <br>Apellido:<input  type="text" name="apellido" size="20"
    onblur="validaCampo(this)">
    <br><input  type="submit" value="enviar">
  </form>
</body>
</html>
```

Fig. 2.45. Código HTML y JavaScript.

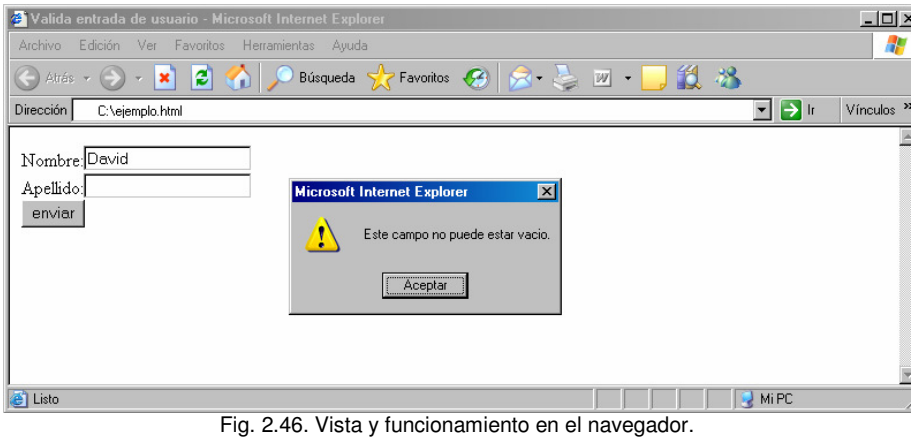


Fig. 2.46. Vista y funcionamiento en el navegador.

También podemos utilizar una función para que, cuando se pulse un botón, se pueda imprimir el contenido de nuestra página Web (figura 2.47).

```
function imprime()
{
  if (confirm("Imprimir?\n"))
  {
    window.print();
  }
}
```

Fig. 2.47. Función que imprime el contenido de la página Web.

Si al llenar un formulario le queremos dar al usuario una posibilidad de revisar si sus datos son correctos, podríamos hacer una función como la mostrada en la figura 2.48.

```
function confirmacion()
{
  if (confirm("Son correctos tus datos?"))
  {
    document.formulario.submit();
  }
}
```

Fig. 2.48. Confirmación de envío de datos.

De igual forma, podemos usar JavaScript para darle un toque agradable a nuestra página como lo muestran los ejemplos de las figuras 2.49 y 2.50.

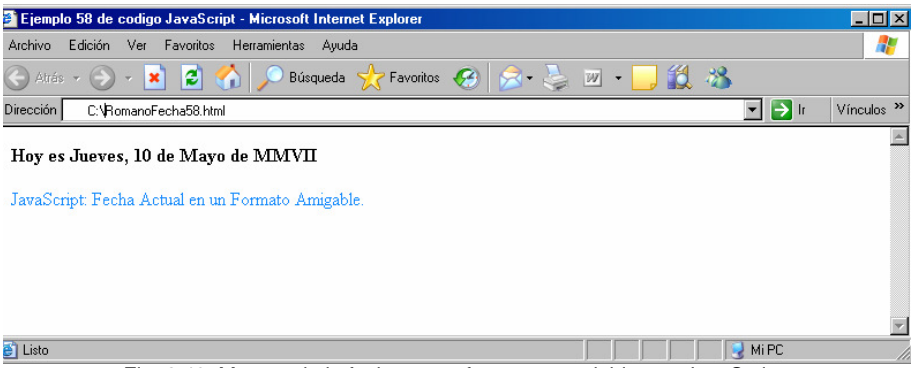


Fig. 2.49. Mostrando la fecha en un formato agradable con JavaScript.

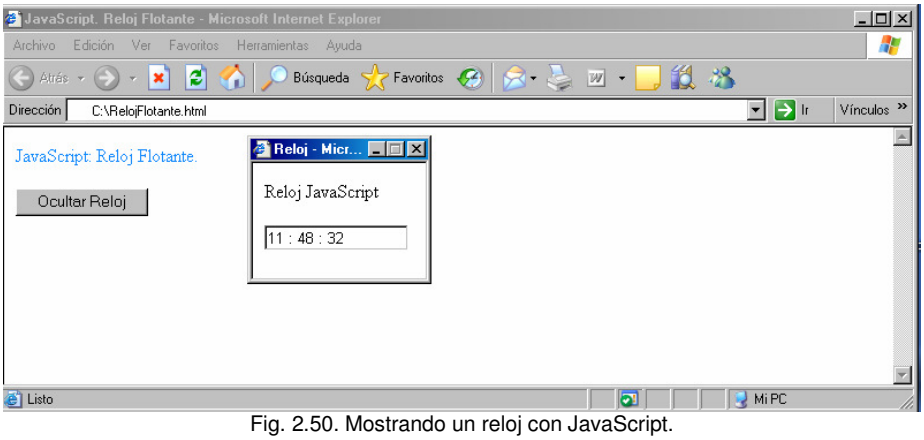


Fig. 2.50. Mostrando un reloj con JavaScript.

CSS

Las hojas de estilo en cascada (*Cascading Style Sheets*, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

La información de estilo puede ser adjuntada en un archivo separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "**style**".

Una ventaja de utilizar CSS (u otro lenguaje de estilo) es que se tiene un control centralizado de la presentación de un sitio Web completo, con lo que se agiliza de forma considerable la actualización del mismo. En las figuras 2.51 y 2.52 podemos ver un ejemplo.

```
<html>
<head>
  <title>Ejemplo CSS</title>
  <script language="javascript">
  </script>
    <style type="text/css">
      <!--
      p.textoingles {text-family:monospace;font-style:italic}
      p.textoespanol {text-family:Verdana,serif;font-size:large}
      .textoenrojo {color:red}
      -->
  </style>
</head>
<body>
   <!-- Parrafos con estilos diferentes -->
   <p class="textoespanol">Este texto esta en español y tiene un estilo distinto al del
     siguiente parrafo, que es en ingles.
   <p class=textoingles>The text is in english and has a different style. Class selectors
     are quite useful at all!
   <p class=textoenrojo>Texto usando una clase no asociada a ninguna directiva concreta!!
   <p class="textoespanol">De nuevo volvemos al español..
   <p class=textoingles>Hola, esto es una prueba
   <h1>Hola</h1>
   <h2 class=textoenrojo></h2>
</body>
</html>
```

Fig. 2.51. Párrafos con estilos diferentes.

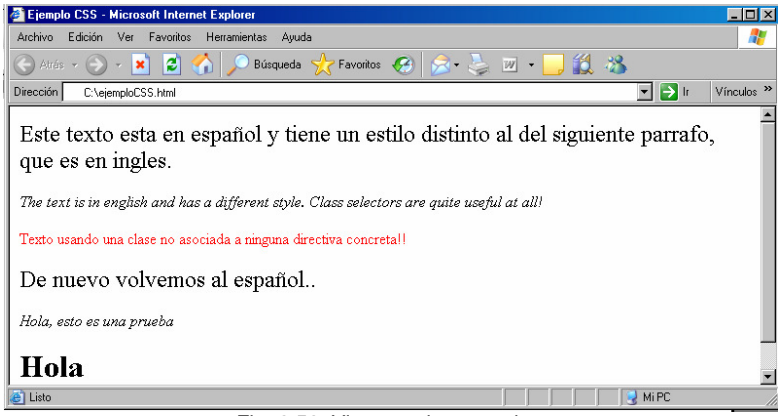


Fig. 2.52. Vista en el navegador.

UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante remarcar que UML es un "lenguaje" para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para soportar una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

Diagramas de estructura. Enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases (figura 2.53).
- Diagrama de componentes.
- Diagrama de objetos.
- Diagrama de estructura compuesta.
- Diagrama de despliegue.
- Diagrama de paquetes.

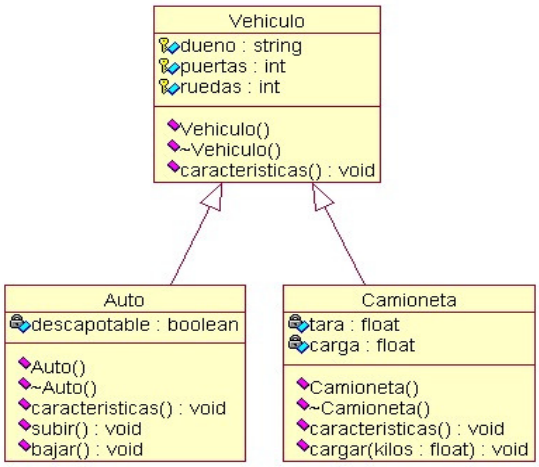


Fig. 2.53. Diagrama de clases.

Diagramas de comportamiento. Enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades.
- Diagrama de casos de uso (figura 2.54).
- Diagrama de estados.

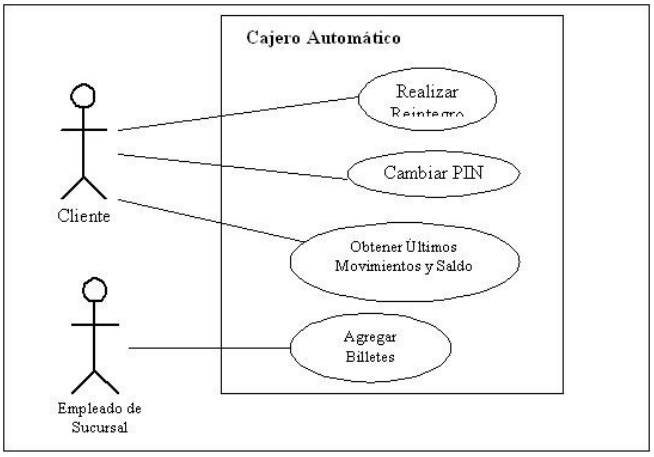


Fig. 2.54. Diagrama de casos de uso.

Diagramas de interacción. Un subtipo de los diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia (figura 2.55).
- Diagrama de comunicación.
- Diagrama de tiempos.
- Diagrama de vista de interacción.

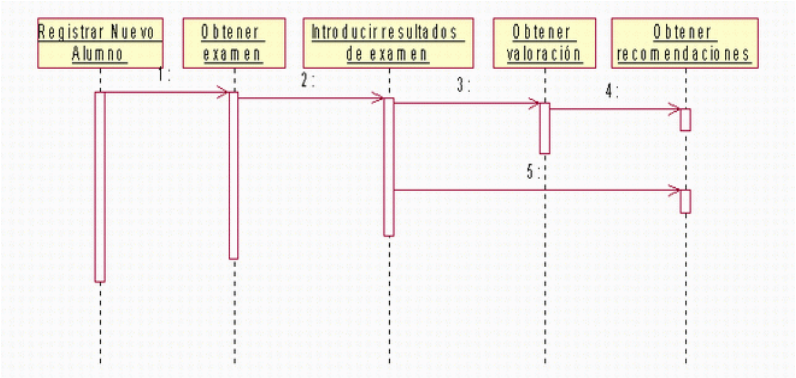


Fig. 2.55. Diagrama de secuencia.

Software libre para modelado en UML: ArgoUML, Dia, gModeler, MonoUML, Papyrus, StatUML, TCM, Umbrello, UMLet.

Herramientas CASE

Desde el inicio de la creación de software ha existido la necesidad de crear herramientas automatizadas que permitan incrementar la productividad de los diseñadores de software, en un inicio, los esfuerzos se direccionaron hacia programas traductores, recopiladores, ensambladores, procesadores de macros, montadores y cargadores.

Al ver los beneficios de este conjunto de aplicaciones se generó una gran demanda por nuevo software con características similares. El significado de las siglas CASE viene de su acrónimo en inglés **C**omputer **A**ided Assisted Automated **S**oftware Systems **E**ngineering.

Las herramientas CASE, en función de las fases del ciclo de vida del desarrollo de sistemas, se pueden agrupar de la siguiente forma:

- **Herramientas integradas**. I-CASE (Integrated CASE): Abarcan todas las fases del ciclo de vida del desarrollo de sistemas, son llamadas CASE workbench.
- **Herramientas de alto nivel**. U-CASE (Upper CASE): Orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo, análisis y diseño.
- **Herramientas de bajo nivel**. L-CASE (Lower CASE): Dirigidas a las últimas fases del desarrollo, construcción e implantación.
- **Juegos de herramientas**. (Tools CASE): Son el tipo más simple de Herramientas CASE, automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

Herramientas CASE más utilizadas

ERwin*:* Es una herramienta para el diseño de base de datos que brinda productividad en su diseño, generación y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada, además, ERwin permite visualizar la estructura, los elementos importantes y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de *stored procedure y triggers* para los principales tipos de base de datos.

DBDesigner: Producto destacable por su sencillez, permite modelar sobre MySQL y dispone de la capacidad de generar documentación e incluso pantallas de administración sobre PHP (figura 2.56).

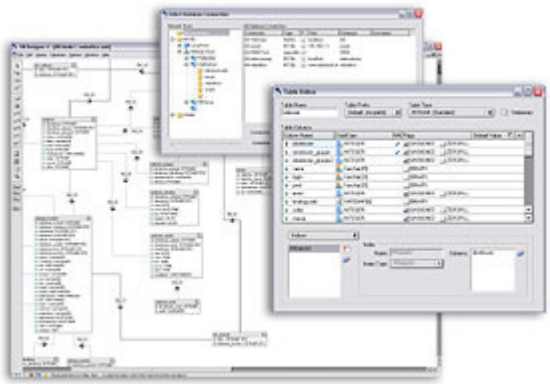
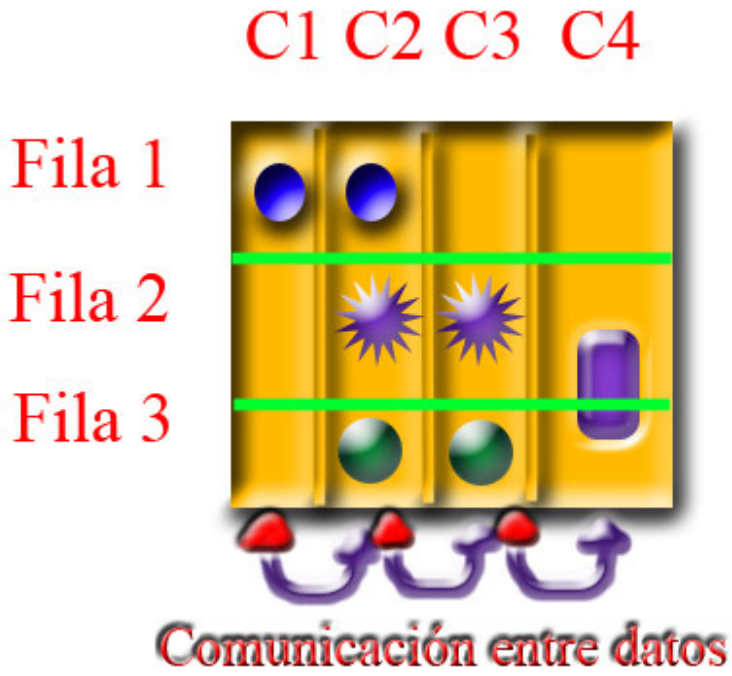


Fig. 2.56. Usando DBDesigner.

CAPÍTULO III
LA BASE DE DATOS



La arquitectura de tres capas se refiere a un diseño que introduce una capa intermedia a los datos y la presentación. Cada una es un proceso separado y bien definido. Estas 3 capas son las siguientes: La capa de presentación (las vistas), la capa de negocios (Modelo y controlador) y la capa de datos (La base de datos).

Entre las ventajas que nos proporciona están la facilidad en su uso, desarrollo y mantenimiento, reducción en costos de migración, etc.

La separación de roles en tres capas, hace más fácil reemplazar o modificar una capa sin afectar a los módulos restantes. Separando la aplicación de la base de datos, hace más fácil utilizar nuevas tecnologías de agrupamiento y balance de cargas.

Este capítulo mostrará que, antes de ayudarnos de herramientas CASE para el diseño y de Postgres o MySQL como manejadores de bases de datos, primero hay que establecer qué datos son los que se desean almacenar y cuál es la mejor forma de hacerlo.

3.1 Aspectos generales

La capa de datos es la base de una aplicación de bases de datos basada en Web, engloba todos los componentes del sistema que almacenan los datos. Esta formada básicamente por el Sistema Gestor de Bases de Datos (Data Base Management System) y un paquete encargado de aislar el acceso a los datos.

La capa de datos es responsable de:

- Almacenar datos
- Recuperar datos
- Mantener datos
- La integridad de los datos

Algunas definiciones de **base de datos**:

- Colección de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos.

- Es una colección de tablas interrelacionadas. El contenido de una base de datos engloba a la información concerniente de una organización, de tal manera que los datos estén disponibles para los usuarios en tiempo real y son compatibles con usuarios concurrentes, una finalidad de la base de datos es eliminar la redundancia o al menos minimizarla. Los tres componentes principales de un sistema de base de datos son el hardware, el software DBMS y los datos a manejar, así como el personal encargado del manejo del sistema.

Las bases de datos son esenciales para el sistema de información de una organización, el cual soporta sus funciones al mantener los datos para éstas y auxilia a los usuarios al interpretar los datos para tomar decisiones.

Los principales problemas que busca eliminar una base de datos son la redundancia y la inconsistencia.

La **redundancia** de datos se refiere, a la existencia de información repetida o duplicada en diferentes tablas dentro de una base de datos.

Frecuentemente los problemas de **consistencia** de datos se deben a la redundancia de éstos. Es muy probable que surjan incongruencias al almacenar la misma información en más de un lugar; ya que al modificar, eliminar o agregar un dato, en esas condiciones, debe realizarse en cada una de las instancias del mismo con el riesgo de no realizarlo en su totalidad, generando en este caso datos inconsistentes.

En este sentido, adquiere importancia el concepto de **integridad** de una base de datos, ésta se refiere no sólo a que los datos sean consistentes dentro de la base, sino además, que los valores que posean los datos sean válidos de acuerdo a las dependencias funcionales entre tablas y de acuerdo a las políticas de negocio.

La integridad de la base de datos se puede lograr mediante el mantenimiento de una redundancia mínima y controlada, el establecimiento de llaves primarias o índices primarios, la validación de las dependencias entre tablas relacionadas y la creación de reglas de validación durante la inserción y edición de datos.

3.2 Modelo de datos

Conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Es una representación de la realidad que contiene las características generales de algo que se va a realizar. En base de datos, esta representación la elaboramos de forma gráfica.

Los modelos de datos se dividen en tres grupos (figura 3.1):

- Modelos lógicos basados en objetos.
- Modelos lógicos basados en registros.
- Modelos físicos de datos.

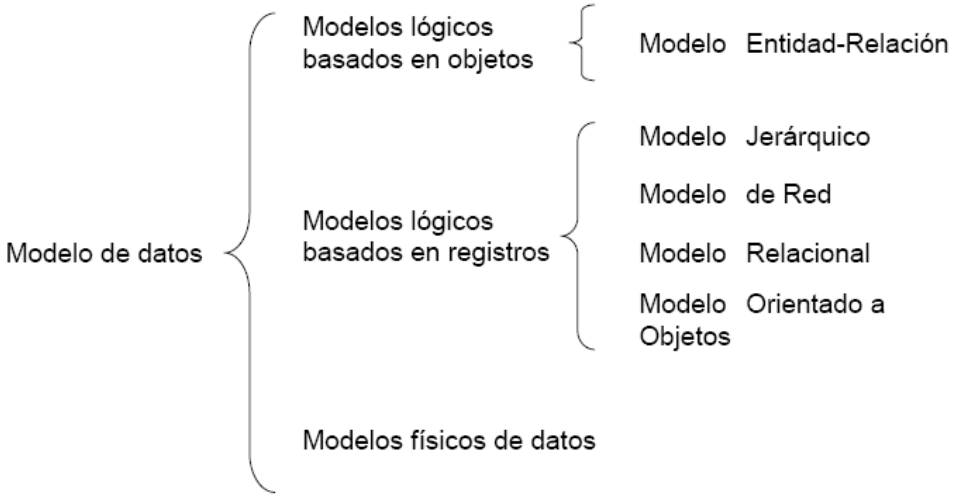


Fig. 3.1. Modelos de datos.

- **Modelos lógicos basados en objetos**

Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación.

- **Modelos lógicos basados en registros**

Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.
Los cuatro modelos de datos más ampliamente aceptados son: Modelo jerárquico, de red, relacional y orientado a objetos.

Modelo relacional

En este modelo se representan los datos y las relaciones entre estos, a través de una colección de tablas, en las cuales los renglones (tuplas) equivalen a cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la tupla.

En el modelo relacional los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las filas y las tablas son visibles en los datos. Este enfoque permite a los usuarios obtener información de la base de datos sin asistencia de sistemas profesionales de administración de información.

Es importante saber que las entradas en la tabla tienen un solo valor (son atómicos); no se admiten valores múltiples, por lo tanto la intersección de un renglón con una columna tiene un solo valor, nunca un conjunto de valores.

Todas las entradas de cualquier columna son de un solo tipo. Cada columna posee un nombre único, el orden de las comunas no es de importancia para la tabla, las columnas de una tabla se conocen como atributos. Cada atributo tiene un dominio, que es una descripción física y lógica de valores permitidos.

No existen 2 filas en la tabla que sean idénticas. La información en las bases de datos son representados como datos explícitos, no existen apuntadores o ligas entre las tablas.

- ***Modelos físicos de datos***

Se usan para describir a los datos en el nivel más bajo, aunque existen muy pocos modelos de este tipo, básicamente capturan aspectos de la implementación de los sistemas de base de datos.

3.3 Bases de datos relacionales

Características:

- Cada “tabla’’ contiene solo un tipo de registros.
- Los campos no tienen un orden específico, de izquierda a derecha.
- Los registros no tienen un orden específico, de arriba hacia abajo.

- Cada campo tiene un solo valor.
- Los registros poseen un campo identificador único (o combinación de campos) llamado clave primaria.

Conceptos:

***Clave primaria**:* Es aquel atributo que identifica de manera única a un registro. Esto es, no debe haber dos tuplas que tengan el mismo valor, por lo tanto, con sólo conocer el valor de la clave primaria para una determinada tupla será suficiente para identificarlo de manera única.

Clave foránea: Es una clave primaria en otra relación, estas representan las asociaciones entre las diferentes entidades, es decir, son claves que están siendo compartidas por dos tablas para formar una relación entre ellas.

Integridad referencial: *"*Si en una relación hay alguna clave foránea, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos*"*. Lo que en realidad trata de decir el texto anterior es que las claves foráneas no pueden dejar de tener correspondencia con la clave primaria de la tabla externa;

Entidades: Se puede definir cono entidad a cualquier objeto, real o abstracto, que existe en un contexto determinado o que puede llegar a existir y del cual deseamos guardar información, por ejemplo, un profesor, un alumno o bien una materia.

Atributos: Las entidades se componen de atributos que son cada una de las propiedades o características que tienen éstas. Cada ejemplar de una misma entidad posee los mismos atributos, tanto en nombre como en número, diferenciándose cada uno de los ejemplares por los valores que toman dichos atributos.

Si consideramos la entidad "Alumno" y definimos los atributos Nombre, Edad, Teléfono y Email, podríamos obtener los ejemplares de la tabla 3.1:

Alumno			
Nombre	**Edad**	**Teléfono**	**Email**
José Francisco Rivera Zarraga	20	99-99-99-99	jfrz@gmail.com
Estrella Mancera Crisóstomo	23	88-88-88-88	emc@yahoo.com
Víctor Alfonso Cruz López	22	77-77-77-77	clva@yahoo.com

Tabla 3.1. Atributos de una entidad.

Grado: Está determinado respecto a la amplitud de la tabla en cuanto al número de atributos.

Dominios: Se define dominio como un conjunto de valores que puede tomar un determinado atributo dentro de una entidad.

Interrelaciones: Se entiende por interrelación a la asociación, vinculación o correspondencia entre entidades. Por ejemplo, entre la entidad "PROFESOR" y la entidad "CURSO" podemos establecer la relación "IMPARTE" por que el profesor imparte cursos.

Asociación 1:1 La ocurrencia de una entidad A se puede relacionar solo con una entidad B y una entidad B puede asociarse únicamente con una entidad A.

Asociación 1:M Se da cuando una entidad se relaciona con cualquier número de ocurrencias en la entidad B, pero una entidad en B puede asociarse únicamente a una entidad A.

Asociación M:M Una entidad en A está relacionada con cualquier número de entidades en B y una entidad en B puede asociarse con cualquier número de entidades en A.

Diccionario de datos: Permite especificar el significado y la composición de los datos, es aplicable a flujos de datos y almacenes (ver ejemplo en el anexo 2).

Normalización
El proceso de cristalización de las entidades y sus relaciones en formatos de tabla usando los conceptos relacionales se llama proceso de normalización (figura 3.2) y consiste en agrupar a los campos de datos en un conjunto de relaciones o tablas que representan a las entidades, sus características y sus relaciones de forma adecuada. La razón de la normalización es asegurar que el modelo conceptual de la base de datos funcionará. Esto no significa que una estructura no normalizada no funcionará, sino que puede causar algunos problemas cuando los programadores de aplicación traten de modificar la base de datos para insertar, actualizar o eliminar datos.

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.
- Está encaminada a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

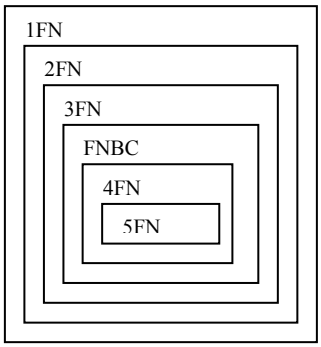


Fig. 3.2. Formas normales.

Primera Forma Normal (1FN)
Una relación está en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna (ver tablas 3.2 y 3.3).

Nombre	Titulaciones
Felipe	Informática Física
Antonia	Administración

Tabla 3.2. No se encuentra en 1FN.

Nombre	Titulaciones
Felipe	Informática
Felipe	Física
Antonia	Administración

Tabla 3.3. En 1FN.

Segunda Forma Normal (2FN)
Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria (tablas 3.4 y 3.5).

NombreProducto	NombreProveedor	Categoría	TeléfonoProveedor
Balón	Niké	Futbol	99999999
Raqueta	Reebok	Tenis	88888888
Zapatos	Adidas	Futbol	77777777
Jersey	Niké	Béisbol	99999999
Guantes	Everlast	Box	55555555

Tabla 3.4. No se encuentra en 2FN.

Idproducto	NombreProducto	Categoría
1	Balón	Futbol
2	Raqueta	Tenis
3	Zapatos	Futbol
4	Jersey	Béisbol
5	Guantes	Box

IdProveedor	NombreProveedor	TeléfonoProveedor
1	Niké	99999999
2	Reebok	88888888
3	Adidas	77777777
4	Everlast	55555555
5	Umbro	44444444

Tabla 3.5. Estas dos tablas ya se encuentran en 2FN.

Tercera Forma Normal (3FN)
Una relación está en tercera forma normal si, y sólo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. La dependencia es transitiva si existen las dependencias siendo atributos o conjuntos de atributos de una misma relación.

La dependencia x→z es transitiva si existen las dependencias x→y, y→z, siendo x,y, atributos o conjuntos de atributos de una misma relación.

Las siguientes tres formas normales casi no se usan y se deja al lector profundizar en ellas:

Forma normal de Boyce/Codd
Para aplicar la forma normal de Boyce/Codd, se deben dar las siguientes circunstancias:

- La relación debe tener dos o más claves candidatas.
- Al menos dos de las claves candidatas deben ser compuestas.
- Las claves candidatas deben tener atributos comunes.
- La forma normal de Boyce/Codd establece, esencialmente, que no debe haber dependencias funcionales entre claves candidatas.

Cuarta forma Normal (4FN)
Se dice que una relación se encuentra en 4FN si y sólo si las únicas dependencias multivaluadas no triviales son aquellas en las que una clave multidetermina un atributo.

No se deben combinar en una sola relación grupos repetitivos independientes.

Quinta forma Normal (5NF)
Se dice que una relación está en 5FN si y sólo si se encuentra en 4FN y toda dependencia de combinación está implicada por una clave candidata.

3.4 Metodología de diseño

El diseño de bases de datos consta de tres etapas:

- Diseño Conceptual
- Diseño Lógico
- Diseño Físico

Diseño conceptual:

1. Identificar las entidades.
2. Identificar las relaciones.
3. Identificar los atributos y asociarlos a entidades y relaciones.
4. Determinar los dominios de los atributos.
5. Determinar los identificadores.
6. Determinar las jerarquías de generalización (si las hay).
7. Dibujar el diagrama entidad-relación.
8. Revisar el esquema conceptual local con el usuario.

El **diseño lógico** es el proceso mediante el que se construye un esquema que representa la información que maneja una empresa, basándose en un modelo lógico determinado, pero independientemente del SGBD concreto que se vaya a utilizar para implementar la base de datos e independientemente de cualquier otra consideración física.

La metodología que se debe seguir para el diseño lógico en el modelo relacional consta de dos fases, cada una de ellas compuesta por varios pasos que se detallan a continuación.

Construir y validar los esquemas lógicos locales para cada vista de usuario

1. Convertir los esquemas conceptuales locales en esquemas lógicos locales.
2. Derivar un conjunto de relaciones (tablas) para cada esquema lógico local.
3. Validar cada esquema mediante la normalización.
4. Validar cada esquema frente a las transacciones del usuario.
5. Dibujar el diagrama entidad-relación.
6. Definir las restricciones de integridad.
7. Revisar cada esquema lógico local con el usuario correspondiente.

Construir y validar el esquema lógico global.

1. Mezclar los esquemas lógicos locales en un esquema lógico global.
2. Validar el esquema lógico global.
3. Estudiar el crecimiento futuro.
4. Dibujar el diagrama entidad-relación final.
5. Revisar el esquema lógico global con los usuarios.

En resumen, la conversión del esquema conceptual a un esquema lógico adecuado al modelo relacional es un paso muy importante y lo que se busca, en general, es:

- Eliminar las relaciones de muchos a muchos
- Eliminar las relaciones complejas
- Eliminar las relaciones recursivas
- Eliminar las relaciones con atributos
- Eliminar los atributos multievaluados
- Reconsiderar las relaciones de uno a uno
- Eliminar las relaciones redundantes.

CAPÍTULO IV
EL MODELO

El Modelo, que forma parte de la capa de negocio, contiene la funcionalidad de la aplicación. Es independiente de las otras dos capas (datos y presentación). Basa sus operaciones en reglas llamadas **reglas de negocio,** que no son más que una colección de políticas y restricciones que controlan el flujo de las tareas. Este capítulo muestra las características más importantes a considerar en la creación de la capa de negocios.

4.1 Aspectos generales

Los servicios de negocios son el enlace entre un usuario y los servicios de datos. En esta capa residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. Cumplen con esto aplicando procedimientos formales y reglas de negocio a los datos relevantes. Cuando los datos necesarios residen en un servidor de bases de datos, garantizan los servicios de datos indispensables para cumplir con la tarea de negocios o aplicar su regla. Esto aísla al usuario de la interacción directa con la base de datos.

El nivel de servicios de negocios es responsable de:

- Recibir la entrada del nivel de presentación (validación de los datos introducidos por el usuario).
- Interactuar con los servicios de datos para ejecutar las operaciones de negocios para los que la aplicación fue diseñada a automatizar.
- Enviar el resultado procesado al nivel de presentación.
- Garantizar la navegación y el flujo de pantallas correcto.

En cuanto a la recepción de la información, se puede hacer de varias maneras, aunque las más habituales son:

- Accediendo a objetos almacenados dentro del Modelo como propiedades.
- El Modelo almacena los resultados como atributos de la petición.
- El Modelo escribe los resultados en la "sesión".

4.2 Características de las reglas del negocio

Desde que nos involucramos con el análisis del negocio, van apareciendo restricciones, políticas, validaciones, fórmulas de cálculo, maneras implícitas de trabajo, suposiciones y declaraciones que forman las denominadas reglas del negocio.

Estas reglas se caracterizan por ser los lineamientos del negocio, la base sobre la cual los procesos logran sus objetivos y se implementan las estrategias del negocio. En muchas ocasiones pueden encontrarse en manuales, códigos, leyes, formatos, reglamentos, pero otras veces son inherentes a la forma de trabajo y caen en tema de suposiciones y no necesariamente deben ser escritas, allí toman el carácter de política de empresa.

El proceso de identificación de estas reglas se desarrolla a medida que el analista del negocio detalle las razones de cada actividad y el por qué de la realización de tal o cuál forma. Los expertos del negocio, los usuarios, conocen muy bien estas reglas y están acostumbrados a ellas, de tal manera que se vuelven parte de su forma de trabajo y será tarea del analista de negocios excavar entre los conocimientos y costumbres de trabajo de estos.

Una de las formas en las que podemos clasificar las reglas de negocio es la siguiente:

Estructura (EST), cálculo (CAL), inferencia (INF), operación (OPE), estímulo y respuesta (E-R), legal (LEG). Y es necesario documentar todas las reglas del negocio identificadas en un **Diccionario de Reglas del Negocio**. Además si es posible deben agruparse por sistemas del negocio de tal manera que se encuentren mejor organizadas. Finalmente la presentación de cada regla del negocio puede documentarse como lo muestra la tabla 4.1:

Nombre:	El saldo de un cliente nunca será menor a $3,000.
Identificador:	RN023
Tipo:	CAL – OPE
Descripción:	El saldo de un cliente tiene que ser controlado y no permitir que sea menor a $3,000. Esto podría general un error en alguna transacción y por lo tanto, el cliente debe ser notificado.
Fuente:	Reglamento de Saldos.
Reglas relacionadas:	RN010, RN022

Tabla 4.1. Ejemplo de regla de negocio.

4.3 Tipos de reglas de negocio

Además de la clasificación hecha anteriormente, otra clasificación de las reglas del negocio en varios grupos es la siguiente:

El primer grupo de reglas de negocio engloba todas aquellas reglas que se encargan de controlar que la información básica almacenada para cada atributo o propiedad de una entidad u objeto es válida: no hay precios de artículos negativos, el sexo de una persona solo puede ser masculino o femenino, una fecha siempre debe ser una fecha válida (por ejemplo, no existe el 30 de Febrero), etc. A estas reglas las llamaremos ***reglas del modelo de datos***.

Otro grupo importante de reglas incluye todas aquellas reglas que controlan las relaciones entre los datos. Estas reglas especifican, por ejemplo, que todo pedido debe ser realizado por un cliente, y que el mismo debe estar dado de alta en nuestro sistema: además, una vez que un cliente haya hecho algún pedido, se deberá garantizar que no es posible eliminarlo, a menos que previamente se eliminen todos sus pedidos. Estas reglas constituyen las ***reglas de relación***.

Es frecuente que a partir de cierta información se pueda derivar otra: por ejemplo, el total de un pedido se puede calcular a partir de las distintas líneas que lo componen, mientras que el total de cada línea se puede calcular a partir del número de unidades vendidas y el precio por unidad. Al conjunto de reglas que especifican y controlan la obtención de información que se puede calcular a partir de la ya existente se las llama ***reglas de derivación***.

Otro grupo de reglas de negocio es el compuesto por las ***reglas de restricción***, que restringen los datos que el sistema puede contener. Nótese que este grupo de reglas se solapa en cierto modo con las reglas del modelo de datos, dado que aquellas también impiden la introducción de datos erróneos, como se vio anteriormente. La diferencia estriba en que las reglas de restricción restringen el valor de los atributos o propiedades de una entidad más allá de las restricciones básicas que sobre las mismas existen: por ejemplo, para un saldo existe una regla básica (regla del modelo de datos) que indica que éste debe ser un número, pero además puede haber una regla que indique que el saldo nunca puede ser menor que cierta cantidad tope establecida para cierto tipo de clientes. Esta sería lo que aquí denominamos una regla de restricción, y la diferencia fundamental estriba en el hecho de que este tipo de reglas requiere para su verificación del acceso a otros fragmentos de información, algo que no sucede con las reglas del modelo de datos.

El último grupo de reglas de negocio incluye aquellas reglas que determinan y limitan cómo fluye la información a través de un sistema. Por ejemplo, un cliente puede hacer una petición de análisis a un laboratorio, que anota un encargado: hecho esto, se genera un parte para uno o más analistas, estos realizan las mediciones correspondientes y devuelven los partes con la información pertinente, a partir de la cuál se genera un informe de análisis, que será un análisis válido solo cuando sea firmado por los responsables de garantizar su corrección. A las reglas que indican qué camino recorre la información y obligan a que se sigan solo los caminos válidos se las llama ***reglas de flujo***.

Suponiendo que tengamos la información en un gestor de bases de datos potente, podremos despreocuparnos de llevar a cabo la codificación de numerosas validaciones en nuestras aplicaciones: así, si en la base de datos creamos una regla de integridad referencial que indica que todo pedido pertenece a un cliente, el gestor de base de datos rechazará cualquier intento de almacenar un pedido en el que se nos haya olvidado indicar el mismo. Cualquier aplicación que acceda a esta base de datos se beneficiará de esta y otras validaciones automáticamente, sin tener que añadir ni una línea de código. Si estamos utilizando una base de datos menos potente, casi todas las reglas de negocio deberán implementarse dentro de los programas que accedan a la base de datos.

4.4 Ubicación de las reglas del negocio

Las *reglas del modelo de datos* especifican los valores válidos de cada atributo de las diversas entidades que se almacenan, lo que simplificando es lo mismo que decir los valores válidos para cada campo de cada tabla. Estas reglas deben, a ser posible, reforzarse en el servidor. Como complemento de esto, sin embargo, se debe implementar estas validaciones también a nivel de cliente, por una simple razón: evitar trabajo y esperas innecesarias a los usuarios.

Por lo que respecta a las *reglas de relación*, el lugar más adecuado para implementarlas es, sin lugar a dudas, el servidor. La mayor parte de los gestores de base de datos proporcionan integridad referencial y los mecanismos necesarios para implementar fácilmente estas reglas, esto proporciona una robustez enorme a la base de datos.

Las *reglas de derivación* pueden variar mucho en complejidad: la información derivada más simple, como calcular el total de una línea de pedido, puede calcularse a partir de otros campos del mismo registro. Dado que no se requiere información adicional, y en general toda la información de un registro viaja a la vez al cliente, calcular esta información en el mismo es trivial, y no resulta gravoso. Ahora bien, si lo que se desea es calcular la suma de todos los pedidos de un cliente (volumen enorme de información), hacerla viajar a través de la red no es recomendable. Lo ideal será implementar el cálculo en el servidor y enviar únicamente al cliente el valor total de la suma.

Las *reglas de restricción* deben implementarse en el servidor. Dado que estas reglas contemplan restricciones en los datos que dependen casi siempre de información presente en varias tablas, llevar a cabo el control en el cliente puede implicar cierto tráfico de red.

Las *reglas de flujo*: dada su complejidad, son implementadas tanto en el cliente como en el servidor.

Por último, vale la pena resaltar la conveniencia de implementar las reglas del modelo de datos también en el cliente, para hacer fluida la interacción con el usuario, siendo lo ideal importarlas dinámicamente del servidor de base de datos.

4.5 El modelo del sistema y el paradigma de la programación orientada a objetos

Un paradigma es una forma de representar y manipular el conocimiento. Representa un enfoque particular o filosofía para la construcción del software. Existen varios (programación funcional, estructurada, dirigida por eventos, orientada a objetos, etc.) y, dependiendo de la situación, un paradigma resulta más apropiado que otro.

La programación orientada a objetos (POO) es un paradigma de la programación que define los programas en términos de "clases de objetos", objetos que son entidades que cambian de estado (datos), comportamiento (métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

En términos orientados a objetos, una opción es encapsular la lógica de negocios en un conjunto de objetos o componentes que no contienen presentación o código de servicios de datos. Se ganará en flexibilidad. Los objetos de negocios son diseñados para reflejar o representar sus negocios. Ellos se convierten en un modelo de sus entidades de negocios e interrelaciones. Esto incluye tanto objetos físicos como conceptos abstractos. Estos son algunos ejemplos de objetos del mundo real: un empleado, un cliente, un producto, una orden de compra, un alumno, un profesor, un curso, una universidad, etc.

Todos estos son objetos en el mundo físico, y la idea en su totalidad detrás de usar objetos de negocios de software, es crear una representación de los mismos objetos dentro de la aplicación. El objetivo es hacer que estos objetos interactúen unos con otros como ellos lo hacen en el mundo real. Un profesor puede impartir un curso a un alumno en alguna universidad.

Cada objeto deberá contener el código necesario para administrarse a si mismo. Un buen diseño de un objeto contiene todos los datos y rutinas necesitadas para representarlo a través del negocio completo, y puede ser usado a través de toda la aplicación de ese negocio.

No toda la lógica de negocio es la misma. Alguna lógica de negocio es un proceso intensivo de datos, requiriendo un eficiente y rápido acceso a la base de datos. Otras no requieren un frecuente acceso a los datos, pero es de uso frecuente por una interfaz de usuario robusta para la validación en la entrada de campos u otras interacciones de usuarios. Si necesitamos una validación al nivel de pantallas y quizás cálculos en tiempo real u otra lógica de negocios, pudiéramos considerar este tipo de lógica de negocios para ser parte de la presentación, ya que en su mayor parte es usada por la interfaz de usuario.

Una alternativa de solución es dividir la capa de lógica de negocios en dos: objetos de negocios de la interfaz de usuario y objetos de negocios de datos.

En realidad, la información puede ser manipulada por muchos programas distintos: así, una empresa puede tener un departamento de contabilidad que controle todo lo relacionado con compras, cobros, etc., y otro departamento técnico, que esté interesado en relacionar diversos parámetros de producción con los costos. La visión que ambos departamentos tendrán de la información y sus necesidades será distinta, pero en cualquier caso siempre se deberán respetar las reglas de negocio.

4.6 Código de ejemplo

```
      VALIDA SI LA CADENA ENVIADA TIENE O NO CARACTERES NUMÉRICOS

<?php
   /* bool cadenaSinNumeros(string) */

   function cadenaSinNumeros($variable)
   {
       for($i=0; $i<strlen($variable);$i++)
       {
           $caracter=substr($variable,$i,1);
           if(is_numeric($caracter))
            {
               $flag=FALSE;
               break;
            }
           else
               $flag=TRUE;
       }
       return $flag;
   }
?>
```

Fig. 4.1. Código de ejemplo 1.

Haciendo uso, de la función predefinida de php is_numeric(), también podemos crear una función propia que nos diga si una cadena tiene caracteres numéricos o no, la figura 4.1 nos lo muestra.

```
  VALIDAR SI HAY O NO UN CORREO ELECTRÓNICO EN LA BASE DE DATOS IGUAL
                  AL QUE SE LE PASA COMO PARAMETRO
<?php

   require("adodb/adodb.inc.php");
   include("librerias/libhtml.inc.php");
   /* bool correoYaExiste(string) */

   function correoYaExiste($email)
   {
       $flag=FALSE;
       $db=conectar();
       $resultado=$db->Execute("SELECT nombre,correo from usuario;");
       while($fila=$resultado->FetchRow())
       {
           if($email==$fila[correo])
           {
               $flag=TRUE;
               break;
           }
       }

       return $flag;
   }
?>
```

Fig. 4.2. Código de ejemplo 2.

La figura 4.2 es una alternativa a la restricción de unicidad (**UNIQUE**) de nuestro manejador de base de datos. Aquí queremos verificar que un dato (en este caso un correo electrónico) no exista en nuestra base de datos antes de intentar insertarlo y, de esta manera, evitar los posibles mensajes de error del manejador y generar uno propio para el usuario.

Podemos comentar lo siguiente: se le dice al intérprete de PHP que necesitará del código de los archivos adodb.inc.php y libhtml.inc.php. Para cargar archivos externos se usan tanto **include** como **require**. Se les llama "construcciones" y sólo difieren en la forma de gestionar los errores. Si se trata de incluir un archivo que no existe, "require" lo considera un error fatal y el programa termina. La construcción "include" es más misericordiosa y simplemente envía un mensaje de error y permite que el programa se siga ejecutando.

Bueno, nos conectamos a la base de datos y se hace una consulta del atributo "correo" de la tabla "usuario", la cual se guarda en la variable resultado. Recorremos registro por registro a través de la función FetchRow() y cada uno es asignado al arreglo llamado "fila" (hemos usado un arreglo sólo para comentar que el atributo "nombre" de cada consulta podría ser referenciado así: $fila[nombre] dentro del mismo ciclo while aunque en está ocasión no se usa y, en lugar de un arreglo, se pudo asignar a una variable normal siempre y cuando la consulta sea de un solo campo de la tabla, en este caso "correo").

Ahora vamos a validar un correo electrónico (figura 4.3), desde el modelo también (capa de negocios). Aunque cabe considerar que esta misma validación pudiera hacerse en una de las páginas Web de la capa de presentación (un formulario) por medio de JavaScript.

VALIDAR UNA DIRECCIÓN DE CORREO ELECTRÓNICO

```
<?php
    .
    .
    if(preg_match('/^[a-z0-9\-\.]+@[a-z0-9\-\.]+\.[a-z]{2,}$/i',$_POST['email']))
    {
        print "correo valido";
    }

    else
    {
        print "correo no valido";
    }
    .
    .
?>
```

Fig. 4.3. Código de ejemplo 3.

- $_POST es una variable de tipo array que alberga los valores de los parámetros de formulario enviado.
- Las funciones PCRE (funciones compatibles con Perl para expresiones regulares) de PHP permiten hacer coincidir una cadena con un patrón y alterar una cadena basado en cómo coincide con un patrón. La función preg_match() comprueba si una cadena coincide con un patrón, se le pasa el patrón y la cadena a comprobar como argumentos. Devuelve 1 si la cadena coincide con el patrón y 0 si no lo hace.
- Un correo valido:
  -Debe tener una @
  -Debe tener delante de la arroba un número cualquiera de caracteres sin tilde, incluyendo números, guiones y puntos, sin espacios en blanco.

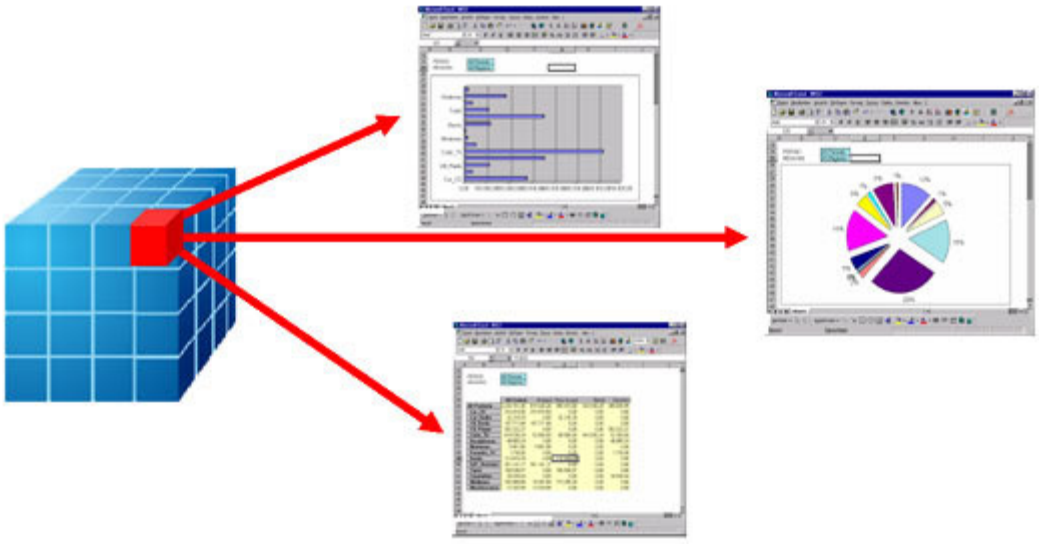
-Tras la @ debe ir el nombre de un dominio, que está formado por dos fragmentos separados por un punto. Delante del punto hay caracteres anglosajones, números, guiones o puntos como sea necesario. Tras el punto, el nombre de dominio debe contener al menos dos letras, que constituyen el dominio de primer nivel (.com, .net, .es, …). Los caracteres de marcado de posición principio (^) y fin de cadena ($) es una restricción de que nuestra cadena es un correo y no que contiene un correo. La opción i nos indica que hará caso omiso de las diferencias entre mayúsculas y minúsculas.

Toda aplicación trata de reflejar parte del funcionamiento del mundo real, para automatizar tareas que de otro modo serían llevadas a cabo de modo más ineficiente, o bien no podrían realizarse. Para ello, es necesario que cada aplicación refleje las restricciones que existen en el negocio dado, de modo que nunca sea posible llevar a cabo acciones no válidas.

La lógica de una interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si realizamos un diseño ofuscado, es decir, que mezcle los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando necesitemos cambiar la interfaz, tendremos que modificar trabajosamente los componentes de negocio. Mayor trabajo y más riesgo de error.

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

CAPÍTULO V
LAS VISTAS



Cada usuario puede tener una vista natural de la misma empresa u organización. Los usuarios pueden ver diferentes conjuntos de objetos como los más trascendentes y dar más importancia a diferentes relaciones. Así, cada uno puede desear la interfaz requerida para resaltar los diferentes aspectos naturales de la base de datos. En este capítulo se identifican los principios y se sugiere una guía para generar una vista adecuada a cada usuario en particular.

5.1 Aspectos generales

Los servicios de presentación proporcionan la interfaz necesaria para presentar información y reunir datos. También aseguran los servicios de negocios necesarios para ofrecer las capacidades de transacciones requeridas e integrar al usuario con la aplicación para ejecutar un proceso de negocios.

La capa de servicios de presentación es responsable de:

- Obtener información del usuario.
- Enviar la información del usuario a los servicios de negocios para su procesamiento.
- Recibir los resultados del procesamiento de los servicios de negocios.
- Presentar estos resultados al usuario.

Una interfaz es creada al idear cualquier tipo de objeto usado por el ser humano, ayudado por la ergonomía, la psicología y el diseño, con la finalidad de que el objeto ideado sea más fácil de usar y mas rápido de aprender.

5.2 Principios

1. Reconoce la diversidad

- **Perfiles de Usuarios**. “Conoce al usuario”, todos los diseños deben comenzar con el entendimiento de los usuarios, incluyendo perfiles de población, genero, habilidades físicas, educación, cultura, entrenamiento, motivación, metas y personalidad. Cada paso en entender al usuario y en reconocerlo como individuo cuyo punto de vista es diferente al del diseñador es un paso más cercano a un diseño exitoso.

  - ✓ **Novatos y primerizos**. Reduce su ansiedad con pocas acciones, una retroalimentación acerca de cada tarea es de mucha ayuda, y cuando el usuario cometa errores deben ser proveídos mensajes de errores específicos y constructivos.

  - ✓ **Usuarios conocedores no frecuentes**. Refresca su memoria con terminología y secuencias de acciones consistentes, ayuda en la pantalla.

  - ✓ **Usuarios expertos y frecuentes**. Dales tiempos de respuesta rápida, retroalimentación breve y discreta. Pon macros y aceleradores.

- **Estilos de interacción**. Cuando el análisis de la tarea está completo, el diseñador puede escoger de estos estilos de interacción:

- ✓ **Manipulación directa**. Interfaces visuales en las cuales el usuario trabaja sobre una representación de los objetos de interés, son extremadamente atractivos, pero se necesita saber cuál es una representación analógica, o metamórfica apropiada.

  Algunos ejemplos son: editores de despliegue y procesadores de palabras, hojas de cálculo, manejadores de datos especiales, video juegos, diseño asistido por computadora, etc.

  Los usuarios que interactúan con este tipo de sistemas deben tener las siguientes características:

  - Expertos en la interfaz.
  - Competencia en hacer la tarea.
  - Facilidad en aprender el sistema y asimilar características avanzadas.
  - Confianza en la capacidad de mantener los conocimientos de la interfaz en mucho tiempo.
  - Disfrutan al usar el sistema.
  - Habilidad para mostrar el sistema a novatos.
  - Deseo de explorar aspectos más poderosos del sistema.

  Pensamiento visual e iconos

  1. Un icono es una imagen que representa un concepto.
  2. Son usados para aprovechar espacio.
  3. Mantente visual para una tarea visual, y mantente en texto para una tarea de texto.
  4. Representa el objeto o acción en una manera familiar y reconocible.
  5. Limita el número de iconos diferentes.
  6. Haz que el icono sobresalga del fondo.
  7. Cuidado con los iconos en 3D, pueden ser llamativos, pero también distraen.
  8. Asegura que un icono seleccionado se vea seleccionado junto con los no seleccionados.
  9. Haz cada icono distinto a los demás.
  10. Asegura que en un grupo de iconos se tenga una armonía.
  11. Diseña una animación para movimiento.
  12. Añade información detallada.
  13. Explora el uso de combinaciones de iconos.

  Ventajas de la manipulación directa

  - Visualmente presentan los objetos.
  - Rápido aprendizaje.
  - Fácil retención.
  - Da paso a la exploración.

  Desventajas

  - Puede ser difícil de programar.
  - Poderoso manejo de gráficos.
  - Poderoso manejo de dispositivos.

- ✓ **Selección de menús**. En estos sistemas, el usuario lee una lista de elementos, selecciona la más apropiada para su tarea y observa su efecto. Aplicaciones de este tipo, tienen un rango de una selección trivial entre dos elementos hasta complejos sistemas de información que ofrecen cientos de despliegues.

Menús simples

- Binarios (figura 5.1).
- Múltiples (Option box).
- Múltiple selección (check box).
- Tipo Pull-down o Pop-up.
- Bidimensionales o navegable.
- Alphasliders.
- Implícitos, ligas en la páginas Web.

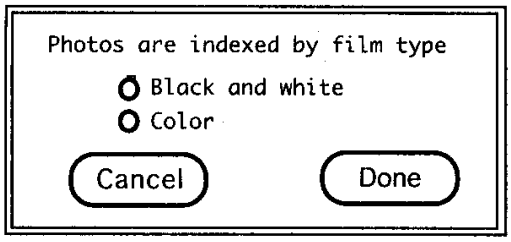


Fig. 5.1. Menú binario.

Secuencias lineales

- Wizards.

Menús de estructura de árbol

- Cajas de diálogo.
- Mapa de una página Web (figura 5.2).

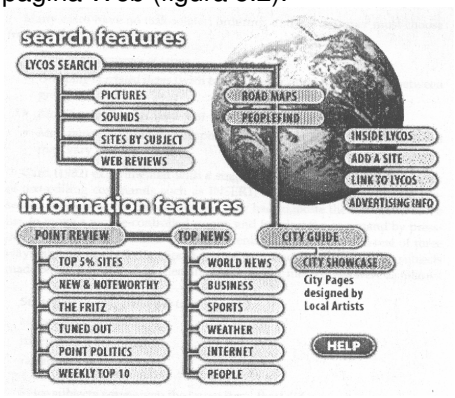


Fig. 5.2. Mapa de una página Web.

Redes cíclicas y acíclicas

- Páginas Web.

Cajas de diálogo. Los usuarios pueden hacer selección de menús, pero algunas tareas requieren también de captura de datos.

El interior

- Título con significado, y estilo consistente.
- Secuencias de arriba izquierda hacia abajo derecha.
- Agrupar y hacer énfasis.
- Consistencia en márgenes, espacios, líneas, cajas.
- Consistencia en terminología, tipografía, justificación.
- Botones estándar (Ok, Cancelar).

El exterior

- Muestra y oculta suavemente.
- Bordes que se distingan pero pequeños.
- Tamaño suficientemente pequeño para reducir problemas de sobrepuesto.
- Despliegue cercano a los elementos apropiados.
- No sobreponer elementos requeridos.
- Fácil de ocultarlo.
- Claridad para completar o cancelar.

Ventajas de la selección de menús

- El aprendizaje es rápido.
- Reduce el número de tecleos.
- Da una estructura de decisión.

Desventajas

- Se puede llenar de menús la pantalla.
- Pueden estorbar a los usuarios frecuentes.

✓ **Llenado de formas**. Cuando se necesita capturar datos, este estilo de interacción es el más apropiado. El usuario ve varios campos, mueve el cursor entre ellos y captura los datos. Los usuarios deben entender las etiquetas de los campos, saber los valores permitidos y el método para capturar los datos.

Ventajas del llenado de formas

- Simplifican la entrada de datos.
- Requiere de poco entrenamiento.
- Permite dar ayuda.

Desventajas

- Consume espacio de pantalla.
- Requiere buena terminología.
- El usuario debe saber los valores permitidos.

✓ **Lenguaje de comandos**. Da una sensación de control e iniciativa. Los usuarios aprenden la sintaxis y pueden expresar posibilidades complejas muy rápidamente. Sin embargo, los rangos de error son muy altos, un entrenamiento es necesario y la retención puede ser pobre.

Ventajas del lenguaje de comandos

- Iniciativa del usuario.
- Permite de creación de macros.

Desventajas

- Pobre manejo de errores.
- Alto entrenamiento y memorización.

- ✓ **Lenguaje natural**. La esperanza de que las computadoras respondan a frases o enunciados arbitrarios ha atraído mucho la atención a desarrolladores, pero su éxito ha sido limitado.

Ventajas del lenguaje natural
- Quita el aprendizaje de sintaxis.
- Ahorra espacio en pantalla.
- Fácil retención.

Desventajas
- Es impredecible.
- Problemas con el contexto.

Mezclar varios estilos de interacción puede ser apropiado cuando la tarea requerida y los usuarios sean diversos.

2. Utiliza las 8 reglas de oro

1. **Consistencia**. Secuencias consistentes de acciones deben ser requeridas en situaciones similares.
2. **Permitir atajos**. Abreviaciones, teclas especiales, comandos escondidos y macros.
3. **Respuestas informativas**. A toda acción del usuario debe existir una respuesta del sistema.
4. **Diálogos con un fin**. Secuencias de acciones deben ser organizadas dentro de grupos con un principio, una parte intermedia y un fin.
5. **Prevención de errores y manejo simple de ellos**. Tanto como se pueda, diseña el sistema de tal forma que el usuario no pueda cometer grandes errores.
6. **Revocación sencilla de acciones**. Tanto como se pueda, las acciones deben ser reversibles.
7. **Soporte de control del sistema**. Operadores experimentados desean sentir que están a cargo del sistema y que el sistema responde a sus acciones.
8. **Reduce la carga de memoria de corto plazo**. La limitación del procesamiento de información en la memoria de corto plazo del humano requiere que los despliegues se mantengan simples.

5.3 Guías

- **Guía para el despliegue de datos**. Smith y Moiser[1] ofrecieron cinco objetivos para el despliegue de datos los cuales hasta ahora siguen siendo vitales:

[1] El informe de Smith y Moiser (ESD-TR-86-278), Agosto de 1986, ofrece directrices para el diseño de software de interfaz de usuario en seis áreas funcionales: entrada de datos, demostración de datos, control de secuencia, dirección de usuario, transmisión de información y protección de datos.

1. **Consistencia**. Durante el proceso de diseño, la terminología, abreviaciones, formatos, colores, deben ser estandarizados y controlados por un diccionario.

2. **Eficiente asimilación**. El formato debe ser familiar para el operador y debe ser relacionado a las tareas requeridas a ser realizadas.

3. **Mínima carga de memoria al usuario**. A los usuarios no se les debe requerir que recuerden información de una pantalla para usar en otra.

4. **Compatibilidad entre el despliegue y la captura de datos**. El formato de información desplegada debe estar claramente ligado con el formato de la captura de datos.

5. **Flexibilidad para que el usuario pueda controlar el despliegue**. Los usuarios deben poder poner la información en el despliegue en la forma en que más le conviene.

Tenemos que considerar que hay información que debe ser presentada de tal forma que atraiga la **atención de los usuarios**. Para ello las siguientes recomendaciones:

- **Intensidad**, no más de dos niveles.
- **Marcar**, subraya, encierra en una caja, apunta con una flecha, o usa un indicador como el asterisco.
- **Tamaño de letra**, no más de 4 diferentes, con tamaños grandes para llamar la atención.
- **Tipo de letra**, no más de 3 distintos.
- **Video inverso**, utiliza un coloreado inverso.
- **Parpadeo**, en áreas limitadas.
- **Color**, hasta 4 colores y adicionales reservados para usos ocasionales.
- **Color cambiante**, utiliza cambios de color (parpadeo entre un color y otro) con cuidado y en áreas limitadas.
- **Audio**, Utiliza tonos suaves para una retroalimentación regular y positiva, y sonidos fuertes para raras condiciones de emergencia.

- **Guía para la captura de datos**. La captura de datos puede ocupar una fracción substancial de tiempo del operador. Smith y Moiser también ofrecieron cinco objetivos de alto nivel para la captura de datos.

1. **Consistencia de transacciones de la captura**. Secuencias similares de acciones deben ser usadas bajo todas las condiciones.

2. **Acciones mínimas de entrada por el usuario**. Menor número de acciones de entrada significan mayor productividad y, a veces, menor posibilidad de cometer errores.

3. **Mínima carga de memoria en los usuarios**. Los usuarios no deben de recordar listas largas de códigos y comandos de sintaxis compleja.

4. **Compatibilidad con la captura y el despliegue**. El formato de entrada debe estar estrechamente ligado a la forma de mostrar la salida.

5. **Flexibilidad para que el usuario controle la entrada de datos**. Usuarios experimentados pueden preferir meter información en una secuencia que ellos controlen. La flexibilidad debe ser usada con cuidado, ya que va en contra del principio de consistencia.

5.4 Evaluación de interfaces

- Factores a tomar en cuenta:

    1. Etapa del diseño (primera, media, última).
    2. Novedad del proyecto (bien definido o exploratorio).
    3. Número de usuarios esperados.
    4. Importancia o tipo de sistema.
    5. Costo del producto y finanzas asignadas para evaluar.
    6. Tiempo disponible.
    7. Experiencia del grupo de diseño y evaluación.

- El rango de evaluación puede ser, dependiendo de la importancia del sistema:

    - Evaluación de varias fases durante dos años.
    - Evaluación de tres días con seis usuarios.

Para evaluar una interfaz se tienen varias formas:

- Revisiones por expertos

    - **Evaluación heurística**, por ejemplo, las 8 reglas de oro.
    - **Guías**, la interfaz es evaluada para que cumplan algunas guías predefinidas para hacer interfaces.
    - **Inspección de consistencia**, los expertos verifican consistencia sobre una familia de interfaces, evaluando consistencia de terminología, color, la forma en que está todo acomodado, formatos de entrada y salida, etc.
    - **Paseo cognitivo**, simulación de usuarios finales usando el sistema.
    - **Inspección formal de uso**, los expertos llaman a una junta con un juez moderador, para presentar la interfaz y discutir sus meritos y sus debilidades.

- Laboratorios

    - Estas pruebas reducen los costos y aceleran los proyectos.
    - Los participantes deben representar a la comunidad de usuarios, con conocimientos en el área y en cómputo.
    - No se debe obligar a nadie y que se sienta libre de irse cuando quiera.

- Pruebas de uso

    - Es muy usado por los diseñadores de juegos.
    - Algunas son del tipo ¿Puedes tronar esto?.
    - Dan mucha seguridad.

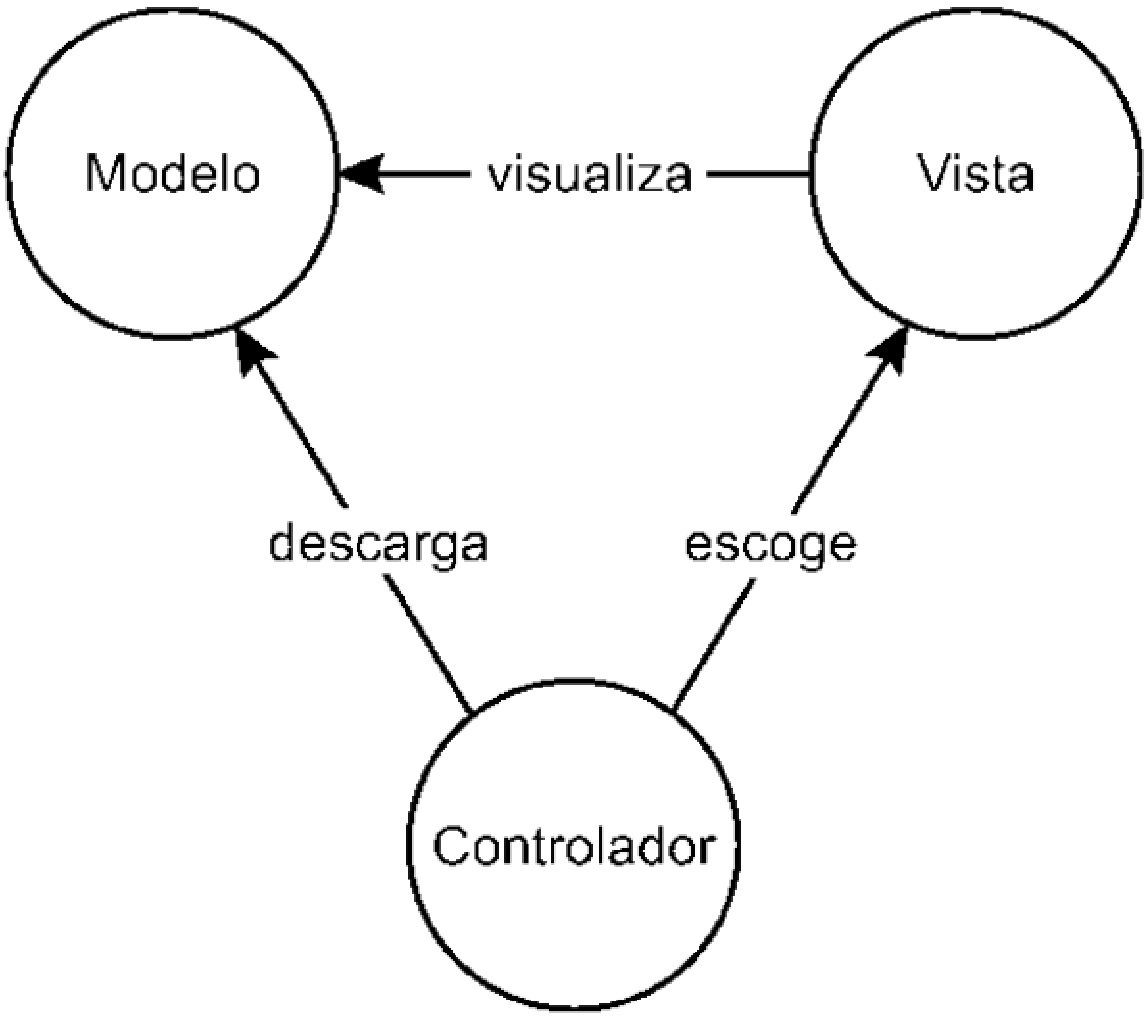
- Laboratorios y pruebas de uso

    - Se enfatiza en usuarios que usan el sistema por primera vez
    - Limita la cobertura de todas las características de la interfaz.
    - Éste y otros problemas obligan a suplementarse con una revisión del experto.

- Cuestionarios. Las claves para tener un cuestionario exitoso es establecer metas claras y enfocarse en elementos que ayuden a alcanzar la meta

En el anexo 3 podemos ver algunos ejemplos de GUIs.

CAPÍTULO VI
EL CONTROLADOR



En este capítulo puntualizaremos las principales tareas de un controlador, independientemente del lenguaje de programación que se utilicé.

6.1 Aspectos generales

Un Controlador, recibe como entrada las peticiones de los usuarios y determina la vista a usar para presentar los resultados.

Cada petición se identifica mediante un parámetro. En base a esta identificación, el Controlador decide qué objeto u objetos de negocio (Modelo) debe ejecutar para resolver la petición.

Tras la ejecución de los objetos de negocio, y en función del resultado devuelto por estos, el Controlador determina qué vista usará para visualizar el resultado, generando una redirección a la página HTML o PHP (en nuestro caso).

El **controlador** es responsable de:

- Recibir los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento X, entonces Acción Y". Estas acciones pueden suponer peticiones al modelo o a las vistas.

6.2 Responsabilidades

Decisiones poco acertadas sobre la asignación de responsabilidades de cada capa y en cada clase, dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender.

Las responsabilidades se relacionan con las obligaciones de un objeto respecto de su comportamiento. Estas responsabilidades pertenecen, esencialmente, a dos categorías: *conocer* y *hacer*.

Entre las responsabilidades de un objeto relacionadas con el ***hacer*** se encuentran:

- Hacer algo en uno mismo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos.

Entre las responsabilidades de un objeto relacionadas con el ***conocer*** se encuentran:

- Estar enterado de los datos privados encapsulados.
- Estar enterado de la existencia de objetos conexos.
- Estar enterado de cosas que se pueden derivar o calcular.

Las responsabilidades se asignan a los objetos durante el diseño. Por ejemplo, podría decirse que una *Venta* es responsable de imprimirse ella misma (un *hacer*), o que una *Venta* tiene la obligación de conocer su fecha (un *conocer*).

Responsabilidad no es lo mismo que método: los métodos se usan para cumplir con las responsabilidades. Éstas se implementan usando métodos que operen solos o en colaboración con otros métodos y objetos. Así por ejemplo, la clase *Venta* podría definir uno o varios métodos para imprimir una instancia *Venta* (por ejemplo el método *imprimir*).

Un patrón útil para la asignación de responsabilidades es el **patrón *Controlador***:

Problema: ¿Quién debería encargarse de atender un evento del sistema?
Un evento del sistema es un evento de alto nivel generado por un actor externo. Es un evento de entrada externa. Se asocia a operaciones del sistema: las que se emiten en respuesta a los eventos del sistema. Por ejemplo, cuando un cajero que usa una terminal de punto de venta oprime el botón "terminar venta", está generando un evento sistémico que indica que "la venta ha terminado". Del mismo modo, cuando alguien que usa un editor de texto pulsa el botón "revisar ortografía", está produciendo un evento del sistema.

Un ***controlador*** es un objeto de interfaz que se encarga de manejar un evento del sistema. Define además el método de su operación.

Solución: Asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase que represente alguna de las siguientes opciones:

- El "sistema" global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo el rol de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso (controlador de casos de uso).

Utilizamos la misma clase Controlador con todos los eventos del sistema en el mismo caso de uso.

Ejemplo:
En la aplicación del punto de venta se dan varias operaciones del sistema, como *terminarVenta(), pasarProducto(), efectuarPago().*

Beneficios: Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la interfaz.

Durante el análisis del comportamiento del sistema, sus operaciones son asignadas al tipo *Sistema*, para indicar que son operaciones del sistema (figura 6.1).

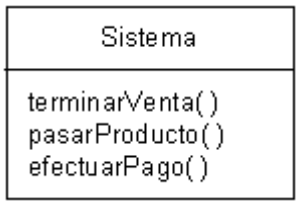


Fig. 6.1. Clase Sistema.

Pero esto no significa que una clase llamada *Sistema* las ejecuta durante el diseño. Durante el diseño, a la clase *Controlador* se le asigna la responsabilidad de las operaciones del sistema.

¿Quién debería ser el controlador de eventos sistémicos como *pasarProducto* y *terminarVenta*? Según el patrón Controlador, disponemos de las opciones que nos muestra la tabla 6.1.

Finalmente, la asignación de responsabilidades está definida en la figura 6.2.

TPDV	Representa el "sistema" global.
Tienda	Representa la empresa u organización global.
Cajero	Representa algo en el mundo real que está activo (por ejemplo el rol de una persona) y que puede intervenir en la tarea.
ManejadordeComprar-Productos	Representa un manejador artificial de todas las operaciones del sistema de un caso de uso.

Tabla. 6.1. Las operaciones del sistema, detectadas en el análisis, se asignarán a *TPDV*.

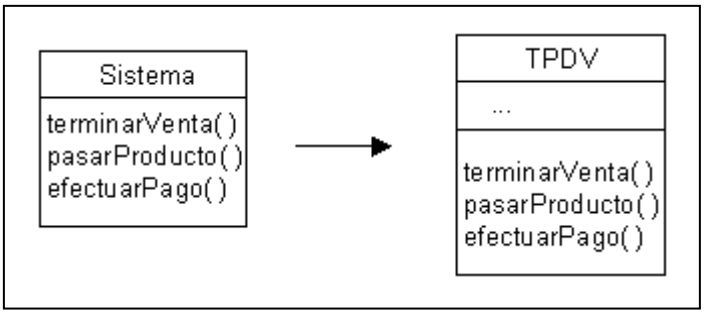


Fig. 6.2. El Controlador de eventos.

La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario (GUI), operada por una persona. Otros medios de entrada son los mensajes externos, o señales procedentes de sensores. En todos estos casos, hay que elegir los controladores que manejan estos eventos de entrada.

La misma clase Controlador debería usarse con todos los eventos sistémicos de un caso de uso. Esto es útil, por ejemplo, para detectar eventos del sistema fuera de secuencia (una operación *efectuarPago* antes de *terminarVenta*, etc).

La primera categoría de controlador, es un controlador de fachada, que representa al "sistema" global. Es una clase que, para el diseñador representa de alguna manera al sistema entero. Si se recurre a la cuarta categoría de controlador (un "manejador artificial de casos de uso"), habrá entonces un controlador para cada caso. Este no es un objeto del dominio, es un concepto artificial.

Por ejemplo, si la aplicación de punto de venta contiene casos como "Comprar Productos" y "Devolver Productos", habrá una clase *ManejadordeComprarProductos* y una clase *ManejadordeDevolverProductos*. Un controlador de casos de uso es una buena alternativa cuando hay muchos eventos de sistema entre varios procesos: asigna su manejo a clases individuales controlables.

Un ejemplo del manejo de muchos eventos es el que nos muestra la figura 6.3, imaginemos una escuela que cuenta con un sistema de información en la que: los alumnos, los profesores, los usuarios externos, los jefes de carrera, etc. tienen acceso restringido a determinada información. Lo más conveniente sería manejar a cada tipo de “actor” por separado, como sigue:

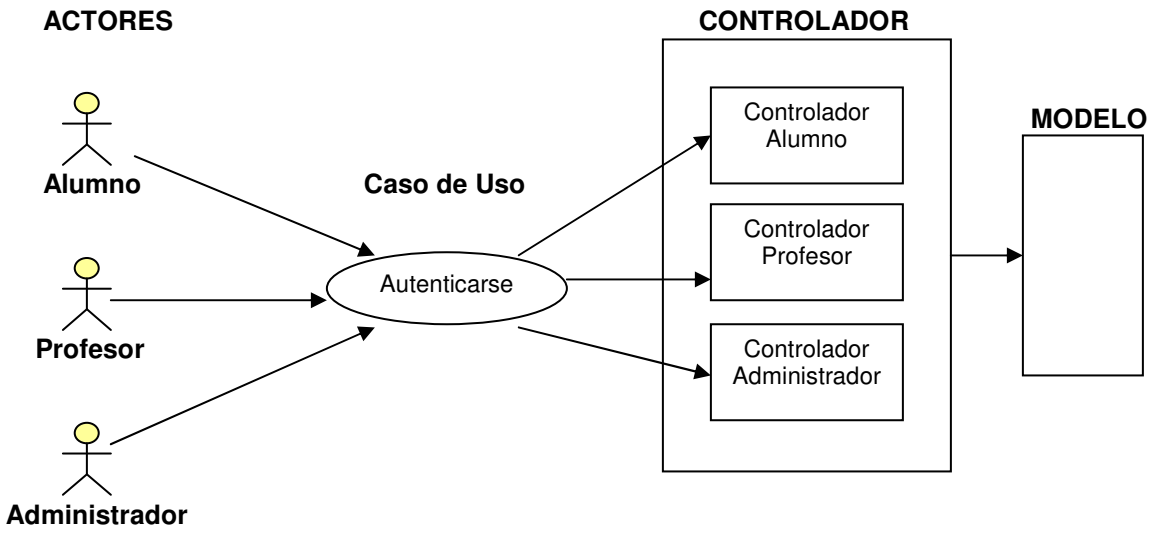


Fig. 6.3. Manejo del evento autenticación.

CONCLUSIONES DEL PROYECTO

CONCLUSIONES

Un sistema de información, independientemente del tipo que sea, contiene toda la información de una organización. Por esta razón es necesario ser muy cuidadosos en su creación, minuciosos con los pequeños detalles y selectivos en cuanto a las alternativas de solución, teniendo en cuenta que la información debe mantenerse segura y al mismo tiempo confiable y flexible.

Respecto a las herramientas utilizadas, el software libre siempre será una opción para el desarrollo de sistemas. Además de reducir costos, nos brinda estabilidad, seguridad y un alto rendimiento que da como resultado la realización de aplicaciones eficientes, mayor control sobre los recursos y la posibilidad de adaptar el software a nuestras necesidades.

No olvidemos que el recurso humano (las personas que interactúan con el sistema) tiene que ser tomado en cuenta en todo el proceso. Un alto porcentaje del éxito de un sistema es la satisfacción de los usuarios, son ellos los que se tienen que sentir cómodos con el sistema. En este caso nuestra tarea será lograr una aceptación del nuevo sistema y después capacitarlos correctamente.

Generalmente, aunque es un tema muy extenso, esto se logra haciendo “sentir” al usuario parte fundamental del cambio, haciendo pruebas de uso con ellos mismos, entrevistas continuas, recabando sus puntos de vista, etc.

En cuanto a las metodologías empleadas, desde la investigación preliminar y la determinación de los requerimientos, es importante reconocer los detalles que pasan inadvertidos para los clientes además de hacer un exhaustivo análisis de la información que se nos proporciona.

En la parte del diseño y desarrollo es conveniente usar patrones que ya existan y adaptarlos a nuestro problema (llámese programación o bases de datos). Recordemos que seguramente alguien ya tuvo un problema parecido al nuestro y que ahora existen procedimientos “comprobados” que lo resuelven.

Las pruebas y la implantación son los dos últimos puntos en el desarrollo de un sistema de información y, aunque aquí no se tocan, la recomendación es que se realicen módulo por módulo para verificar en cada paso que todo está funcionando bien. Cuando el sistema está integrado por completo, una técnica es probar en paralelo (al mismo tiempo que el sistema anterior para comprobar resultados). Nunca deben hacerse pruebas mientras el sistema se encuentre en producción.

La última parte, aunque no menos importante, es hacer una documentación clara del sistema, tanto para su uso como para darle mantenimiento y analizar posibles ampliaciones.

En general, una buena práctica es utilizar todos los pasos de la ingeniería de software y calendarizar nuestras actividades.

GLOSARIO

ADODB: viene de "Active Data Objects DataBase" Es una librería de abstracción de los datos que asegura en gran medida la portabilidad de una aplicación que necesite comunicarse con bases de datos.

Aplicación Web: Consiste básicamente en generar documentos HTML dinámicos, por "dinámicos" se debe entender que dichos documentos de HTML no existen como tal en el servidor Web sino que se generan en el momento en que son solicitados.

Base de datos: Colección de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real.

CGI: La tecnología CGI "Common Gateway Interface" define un modelo de programación que puede ser implementado por múltiples lenguajes. Los CGIs son programas que siguen un estándar definido, corren en el servidor, reciben parámetros desde el cliente y su salida es enviada al navegador.

Comando: Medio por el cual se le ordena una acción determinada al sistema operativo a través de un intérprete.

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

Controlador: Componente del MVC que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Criptografía: Del griego kryptos, "ocultar", y grafos, "escribir", literalmente "escritura oculta" es el arte o ciencia de cifrar y descifrar información utilizando técnicas matemáticas que hagan posible el intercambio de mensajes de manera que sólo puedan ser leídos por las personas a quienes van dirigidos.

CSS: Las hojas de estilo en cascada (*Cascading Style Sheets*,) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

Datos: Son flujos hechos en bruto que representan sucesos ocurridos en las organizaciones o en el entorno físico, antes de ser organizados y acomodados de tal forma que las personas puedan entenderlos y usarlos.

DBMS: DataBase Management System, su expresión inglesa. Equivalente de SGBD (Sistema Gestor de Bases de Datos). Son sistemas de gestión de base de datos, un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Demonio: En Unix/Linux se conoce como un programa que permanece en segundo plano ejecutándose continuamente para dar algún tipo de servicio. Ejemplos de demonio, son los servidores de correo, impresora, sistemas de conexión con redes, etc.

Distribución: Un sistema operativo (en general Linux), que se ha empaquetado para facilitar su instalación.

Eficiencia: Capacidad administrativa de producir el máximo de resultados con el mínimo de recursos, el mínimo de energía y en el mínimo de tiempo posible.

Expresión Regular: Conjunto de caracteres que forman una plantilla para buscar y reemplazar cadenas de texto dentro de textos más largos.

Framework: En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otras herramientas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

FSF: Free Software Foundation. Fundación que pretende el desarrollo de un sistema operativo libre tipo Unix. Empezó creando las herramientas necesarias para su propósito, de modo que no tuviera que depender de ninguna compañía comercial.

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

GNOME: Es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Windows.

GPG: Gnu Privacy Guard (GnuPG), es una herramienta para cifrado y firmas digitales, es software libre licenciado bajo la GPL.

GPL: General Public License. Una de las mejores aportaciones de la FSF. En la Licencia Pública General el autor conserva los derechos de autor (copyright) y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que no sea imposible crear un producto con partes no licenciadas GPL: el conjunto tiene que ser GPL.

GUI: (Interfaz gráfica de usuario). Es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

Herramientas CASE: (**C**omputer **A**ided **S**oftware **E**ngineering). Ingeniería de Software Asistida por Computadora, son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero.

Heurística: manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

HTML: Acrónimo inglés de **H**yper**T**ext **M**arkup **L**anguage (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

HTTP: Es el protocolo de red para el WWW, basa su operación en la arquitectura "Cliente-Servidor". El servidor HTTP es el encargado de publicar "recursos" electrónicos y el cliente consulta los recursos que el servidor ofrece.

Implementar: Poner en funcionamiento, aplicar métodos, medidas, etc., para llevar algo a cabo.

Información: Se refiere a los datos a los que se les ha dado una forma que tiene sentido y es útil para los humanos.

Integridad referencial: Propiedad de las bases de datos que garantiza que un registro se relacione con otros registros válidos.

IP: Las direcciones IP (Internet Protocol) son el método mediante el cual se identifican los ordenadores individuales (o, en una interpretación más estricta, las interfaces de red de dichos ordenadores) dentro de una red TCP/IP.

KDE: (**K D**esktop **E**nvironment) es un entorno de escritorio e infraestructura de desarrollo para sistemas Unix y, en particular, Linux.

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

Licencia MIT: Esta licencia permite reutilizar el Software, tanto para ser libre como para ser software no libre, permitiendo no liberar los cambios realizados al programa original. También permite licenciar dichos cambios con licencia BSD, GPL, u otra cualquiera que sea compatible (es decir, que cumpla las cláusulas de distribución). Con esta licencia se tiene software libre.

Licencia BSD: El autor, bajo esta licencia, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación.

Librerías: Se refiere al conjunto de rutinas que realizan las operaciones usualmente requeridas por los programas.

Linux: Sistema operativo compuesto de las herramientas GNU de la FSF y el núcleo desarrollado por Linus Torvalds[1] y sus colaboradores.

Metodología: Se refiere al conjunto de los pasos y métodos sistemáticos de investigación en una ciencia, que guían o deberían guiar una investigación

Modelo: Componente del MVC. Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación, es otra forma de llamar a la capa de dominio.

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

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

Modelo Vista Controlador (MVC): es un patrón de arquitectura de software (de tres niveles) que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Patrón de diseño: Es una solución a un problema de diseño no trivial que es efectiva y reusable, es decir, se puede aplicar a diferentes problemas de diseño en distintas circunstancias.

Programación orientada a objetos: La POO es un paradigma de programación que define los programas en términos de "clases de objetos". Se trata de una filosofía de programación donde la realidad se modela mediante objetos y la interacción entre ellos.

Políticas: Son guías para orientar la acción; son criterios, lineamientos generales a observar en la toma de decisiones, sobre problemas que se repiten una y otra vez dentro de una organización.

Proceso: Programa en ejecución dentro de un sistema informático.

[1] Linus Benedict Torvalds (nacido el 28 de Diciembre de 1969 en Helsinki), es un ingeniero de software finlandés.

Reglas del negocio: Es el conjunto de restricciones, políticas, validaciones, fórmulas de cálculo, maneras implícitas de trabajo, suposiciones y declaraciones que forman parte de una organización.

Root: Persona o personas encargadas de la administración del sistema.

Servidor Web: Es un programa que se ejecuta en segundo plano sirviendo documentos Web a petición de los clientes (navegadores) que se lo piden.

Sistema de información: Conjunto de componentes o procedimientos interrelacionados que obtiene, procesa, almacena y distribuye información para apoyar la toma de decisiones, la coordinación y el control en una organización. También ayudan a los administradores y trabajadores a analizar problemas, visualizar aspectos complejos y crear productos nuevos.

Sistema Operativo: (S.O.) Es un conjunto de programas de control que tiene por fin facilitar el uso de la computadora y conseguir que ésta se utilice eficientemente.

Shell: Es el intérprete de comandos que se establece entre nosotros y el kernel.

Software libre: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

SQL: El Lenguaje de Consulta Estructurado (**S**tructured **Q**uery **L**anguage) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

Swap: Espacio de disco duro que utiliza el Kernel en caso de necesitar más memoria de la que tengamos instalada en nuestro ordenador.

Triggers (Disparadores): Un disparador es un objeto con nombre dentro de una base de datos, el cual se asocia con una tabla y se activa cuando ocurre en ésta un evento en particular.

Tubería: Las tuberías (filtros) son conexiones entre procesos. La salida de un proceso la encadenamos con la entrada de otro.

UML: Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Vista: Componente del MVC que presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.

WWW: World Wide Web. Sistema con estándares universales aceptados para almacenar, recuperar, formatear y exhibir información de un ambiente de red.

BIBLIOGRAFÍA

- Alarcón, José Manuel. “JavaScript”. Anaya Multimedia Madrid 2004.
- Álvarez García, Alonso. “HTML 4.1”. Anaya Multimedia Madrid 2003.
- Blanco, Vicente J. “Linux: Instalación, administración y uso del sistema”. Alfaomega México 1997.
- Bobadilla, Jesús. “Html dinámico a través de ejemplos”. Alfaomega Colombia 2000.
- Carling, M. “Guía avanzada de Administración de Sistemas Linux”. Prentice-Hall Madrid 2000.
- Castaño, Adoración de Miguel. “Diseño de Bases de Datos: problemas resueltos”. Ra-Ma Madrid 2001.
- Galeano, Germán. “Manual imprescindible de html 4”. Anaya Multimedia Madrid 2000.
- Gutiérrez, Abraham. “PHP4 a través de ejemplos”. Alfaomega México 2004.
- Gutiérrez Gallardo, Juan Diego. “MySQL 5”. Anaya Multimedia Madrid 2006.
- Hawryszkiewycz, I.T. “Analisis y Diseño de Bases de Datos”. Noriega Editores México 1994.
- Ibarra Obando, Alma. “Páginas para Internet”. UNAM: DGSCA México 2000.
- Kabir, Mohammed. “Servidor Apache”. Anaya Multimedia Madrid 1999.
- Laudon, Kenneth C. “Sistemas de Información Gerencial”. Pearson Educación México 2002.
- Leblanc, Dee-Ann. “La Biblia de administración de sistemas Linux”. Anaya Madrid 2001.
- López Camacho, Vicente. “Linux guía de instalación y administración”. McGraw-Hill Madrid 2000.
- Martín Ramos, Monso. “Software Libre para sitios Web”. MP Ediciones Buenos Aires, Argentina 2004.
- Pérez, Cesar. “Desarrollo de páginas Web Dinámicas con PHP y MySQL“. Alfaomega México 2004.
- Sánchez Prieto, Sebastián. “Unix y Linux guía práctica”. Alfaomega México 2000.
- Shneiderman, Ben. “Diseño de interfaces de usuario: estrategias para una interacción persona-computadora efectiva”. Pearson Educación Madrid 2006.
- Sklar, David. “Introducción a PHP 5”. Anaya Multimedia Madrid 2005.
- Soria Momparler, R. “Navegar en Internet: diseño y creación de páginas web html”. Alfaomega México 1999.
- Steve, Shan. “Manual de administración de Linux”. McGraw-Hill Madrid 2000.
- Trigos García, Esteban. “PHP 4”. Anaya Madrid 2000.

REFERENCIAS

- http://adodb.sourceforge.net/
- http://www.apache.org/
- http://www.mysql.com/
- http://www.php.net/
- http://www.postgresql.org/

ANEXOS

Anexo 1. Comandos básicos de vi

vi es un editor de texto interactivo, diseñado para utilizarse con un terminal provisto de pantalla. En dicha pantalla se muestra normalmente una "ventana" del fichero que se está editando; esta última permite ver unas 20 líneas del fichero al tiempo, y además la ventana se puede subir o bajar a lo largo del fichero. También se puede trasladar a cualquier parte de cualquier línea de la pantalla, y realizar cambios allí; las adiciones y cambios que se realizan en el fichero quedan reflejados sobre lo que se ve en pantalla. Las siglas vi de este editor se derivan de *"visual"*.

Modo de inserción: Insertar texto en el archivo.
Modo de pantalla: Mover el cursor dentro de la pantalla.
Modo de última línea: Comandos desde la última línea de pantalla.

Insertar texto

i	→	Antes del cursor.
a	→	Después del cursor.
I	→	Al inicio de la línea.
A	→	Al final de la línea.
o	→	Abre una línea abajo del cursor.
O	→	Abre una línea arriba del cursor.

vi archivo	→	Abre el archivo para capturar más información.
vi	→	Capturar en archivo nuevo.

Cambiar

cw	→	Cambia una palabra.
cc	→	Cambia una línea.
C	→	Cambia desde el cursor hasta el final de la línea.
r	→	cambia el carácter donde se encuentra el cursor.
R	→	Cambia el texto desde donde se encuentra el cursor.

Borrar

dw	→	Borra una palabra.
dd	→	Borra una línea 10dd→ Borra 10 líneas.
x	→	Borra el carácter donde se encuentra el cursor.
X	→	Borra el carácter a la izquierda del cursor.
D	→	Borra desde el cursor hasta el final de la línea.

Última Línea

:w nombre_de_archivo	→	Graba el archivo.
:q	→	Sale de "vi" sin guardar.
:wq	→	Guarda el archivo y sale de "vi".
:n	→	Siguiente archivo.
:r	→	Leer archivo.
:e	→	Editar un archivo.
:f	→	Muestra el nombre del archivo.
:se un	→	Muestra números de línea en el archivo.
:se nonu	→	Oculta los números de línea del archivo.
:.=	→	Indica el número de línea donde se encuentra el cursor.
:=	→	Muestra el número total de líneas que tiene el archivo.

Otras Funciones

u	→	Deshacer los cambios en la línea.
/	→	Busca una cadena hacia delante.
?	→	Busca una cadena hacia atrás.
n	→	Siguiente ocurrencia.
N	→	Ocurrencia anterior.
.	→	Repite la última acción.
nY	→	Copia “n” líneas a partir del cursor.
P	→	Coloca la línea copiada arriba del cursor.
p	→	Coloca la línea copiada abajo del cursor.
ZZ	→	Graba el archivo y sale del editor.

Movimiento del cursor

l	→	Un espacio a la derecha.
h	→	Un espacio a la izquierda.
j	→	Una línea hacia abajo.
k	→	Una línea hacia arriba.
$	→	Mueve el cursor al final de la línea.
w	→	Mueve el cursor a la siguiente palabra.
e	→	Mueve el cursor al final de la siguiente palabra.
b	→	Mueve el cursor al inicio de cada palabra previa.
nG	→	Mueve el cursor a la línea “n”.
G	→	Mueve el cursor a la última línea del archivo.
-	→	Al inicio de la línea anterior.
+	→	Al inicio de la línea siguiente.
n|	→	Mueve el cursor al a columna “n”.
H	→	Mueve el cursor a la línea superior en la pantalla.
M	→	Mueve el cursor al a mitad de la pantalla.
L	→	mueve el cursor a la última línea de la pantalla.
^D	→	Avance hacia delante media pantalla.
^U	→	Avance hacia atrás media pantalla.
^F	→	Siguiente pantalla.
^B	→	Pantalla anterior.

Nota: Dentro de vi, si se quiere hacer alguna modificación primero se teclea <esc>, después nos colocamos en la palabra o línea correspondiente y por último se teclea el comando que se desee.

Anexo2. Ejemplos de diccionario de datos

Simbología

=	Compuesto de
+	Y
()	Opcionalidad
{}	Iteración
[|]	Selección de alternativas
* *	Comentario
“ ”	Contenido textual (no estándar)
-	Rango (no estándar)
@	Clave de acceso

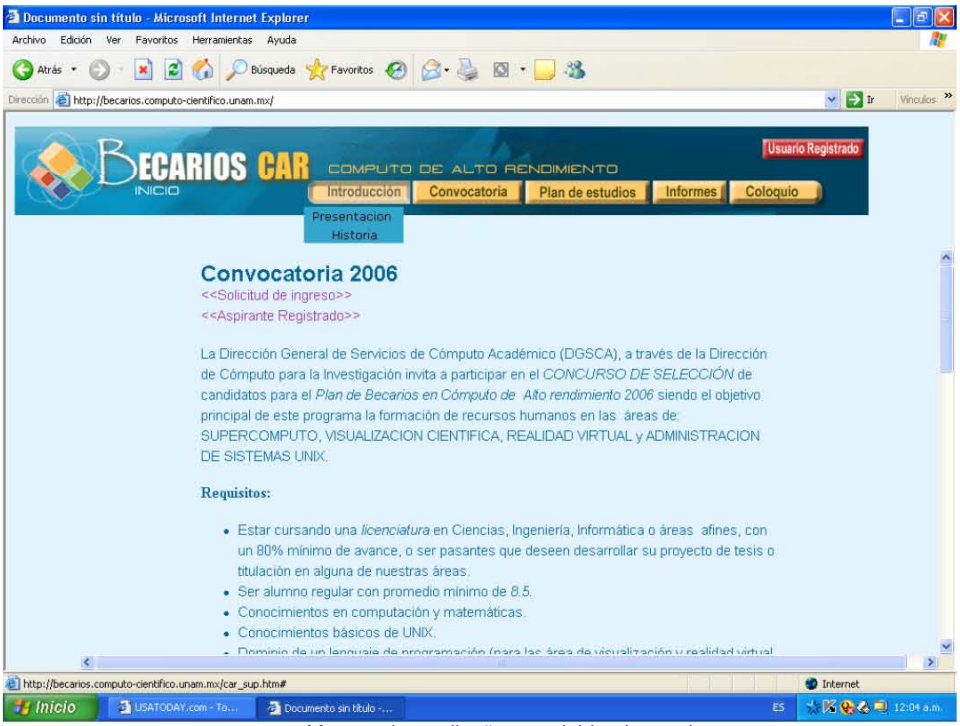
Ejemplo 1

Nombre = TituloCortesia
+ NombrePila
+ Apellido
TituloCortesia = ["Sr" | "Sra"]
NombrePila = { Carácter }
Apellido = { Carácter }
Carácter = ["A"-"Z" | "a"-"z"]

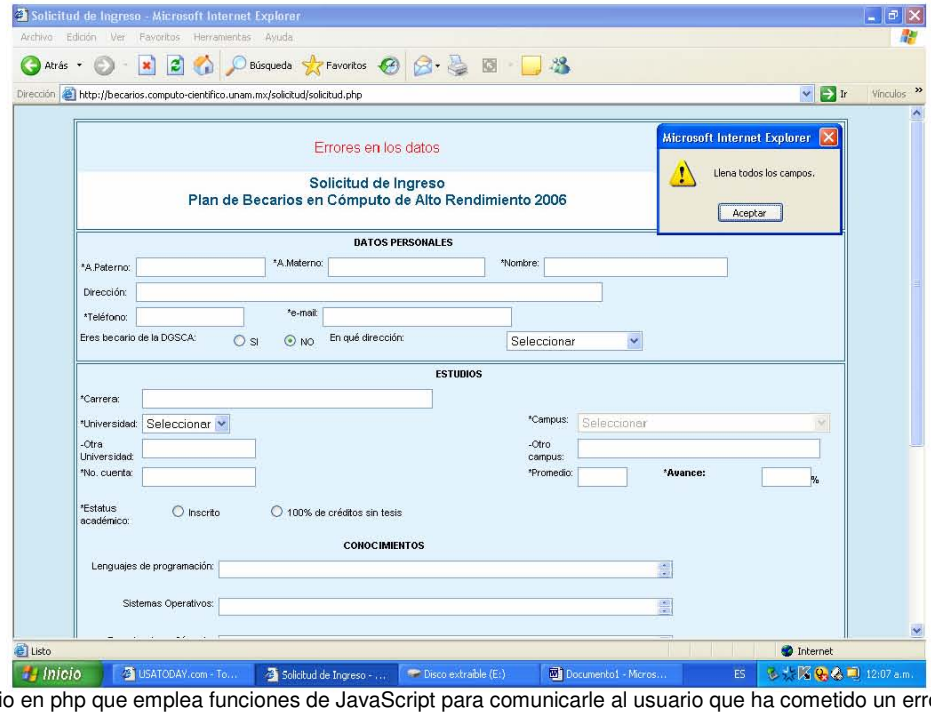
Ejemplo 2

Fecha Pedido	* Fecha en que se realiza el pedido * Por defecto: Fecha del sistema
Fecha Prevista Entrega	* Fecha prevista para la entrega del pedido * Por defecto: Una semana después de la fecha del sistema Rango: [Fecha del sistema, Fecha del sistema + 7 días naturales]
Cantidad Pedida	* Cantidad de material pedida * Por defecto: 10 Unidades: Toneladas Rango: [1, 50]

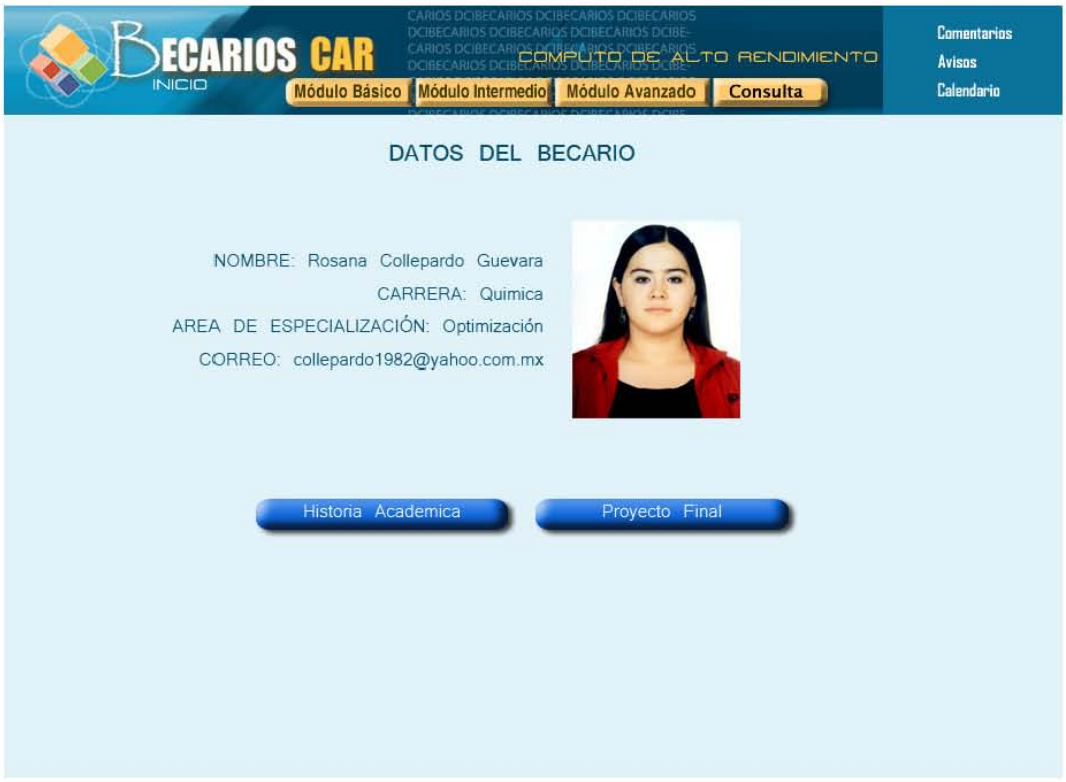
Anexo 3. Ejemplos de GUIs de un sistema real creado con software libre



Mostrando un diseño agradable al usuario



Formulario en php que emplea funciones de JavaScript para comunicarle al usuario que ha cometido un error en la captura



Obteniendo información de la base de datos para mostrarla al usuario

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS