UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES-CUAUTITLAN

DEPARTAMENTO DE EXAMENES PROFESIONALES

"SISTEMA DE SEGUIMIENTO A EGRESADOS DE LA FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN"

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN INFORMATICA

P R E S E N T A :

RICARDO PEREZ GARCIA

ELVIA SANCHEZ MAGADAN

ASESOR: L.C. CARLOS PINEDA MUÑOZ

CUAUTITLAN IZCALLI, EDO. DE MEX 2002.

TESIS CON
FALLA DE ORIGEN

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

ASUNTO: VOTOS APROBATORIOS

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES-CUAUTITLAN

DEPARTAMENTO DE EXAMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

Sistema de seguimiento a egresados de la Facultad de Estudios Superiores Cuautitlán.

que presenta la pasante: Elvia Sánchez Magadán
con número de cuenta: 9555955-1 para obtener el título de :
Licenciada en Informatica

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 11 de Marzo de 2002

PRESIDENTE	Ing. Sergio Pedro Acosta Torres
VOCAL	L.C. Carlos Pineda Múñoz
SECRETARIO	Ing. Víctor H. Arroyo Hernández
PRIMER SUPLENTE	Lic. Conrado Camacho Arteaga
SEGUNDO SUPLENTE	L.C. Jaime Navarro Mejía

DEDICATORIAS

A mis padres:

Que no tengo manera de darles las gracias por todo el apoyo que me brindaron, el cariño y la confianza que siempre me han tenido.

A mis hermanos:

Por ser una guía de enseñanza para lograr este objetivo.

A Daniel:

Que me ha enseñado que todo se puede lograr en esta vida y con él que siempre he contado.

A Elvia:

Por todo el apoyo brindado, el cariño, y sobretodo que me ha enseñado a valorar las cosas buenas y malas de la vida, para sacarles el mejor provecho, y por todos los consejos que han hecho ser una mejor persona.

A L.C. Carlos Pineda Muñoz:

Por todo el apoyo brindado para culminar este proyecto.

A la UNAM:

Por todos los conocimientos recibidos.

RICARDO PÉREZ GARCÍA

A mis padres:

Sabiendo que jamás existirá una forma de agradecer el apoyo, la lucha y el cariño. Deseo expresarles que los esfuerzos y logros alcanzados han sido también suyos. Y forman parte del éxito más grande que he podido recibir. Con amor para ustedes.

A mis hermanos y cuñado:

A ustedes por el apoyo y confianza. Y en especial a una mujer a quien admiro y quiero, quien siempre confió en mi, para ti Ara.

Y por el compromiso que tengo con mis niños Brayan, Brenda, Yatziry, Moncerrat, Andrea y Yessica, que tanto amo y que siempre pueden contar conmigo.

A Ricardo:

Por el apoyo, comprensión y confianza que me ayudo hacer menos pesada esta vida de lucha y superación constantes, con cariño y admiración para ti.

A mi Asesor de Tesis:

Por el apoyo y el tiempo aportado para la terminación del proyecto de titulación.

ELVIA SÁNCHEZ MAGADÁN

INDICE

CONTENIDO

Introducción 1

Objetivos 2

Alcance del Sistema 2

Herramientas de Desarrollo 4

Resultados Esperados 6

Capitulo 1. CONCEPTOS DE ANÁLISIS Y DISEÑO DE SISTEMAS 8

1.1 Introducción 8

1.1.1 Breve Revisión Histórica 10

1.1.1.1 Evolución de los Lenguajes de Programación 11

1.1.2 ¿Qué es la POO? 16

1.1.3 Programación Orientada a Objetos 17

1.1.3.1 Conceptos Básicos 17

a) Definición de Clase 18

b) Definición de Objeto 19

c) Abstracción 19

d) Encapsulación 20

e) Polimorfismo 21

f) Herencia 21

g) Mensaje 24

h) Sobrecarga 25

1.1.4 Características de los Lenguajes Orientados a Objetos 25

1.1.5 Beneficios de las Tecnologías de Objetos 27

1.1.6 Limitaciones e Inconvenientes de la OOP 27

1.2 UML (Lenguaje Unificado de Modelado) 30

1.2.1 Introducción 30

1.2.2 Elementos de UML 31

1.2.2.1 Diagramas de UML 31

a) Diagrama de Clases 31

b) Diagrama de Objetos 35

c) Diagrama de Casos de Uso 36

d) Diagrama de Estados 39

e) Diagrama de Secuencias 41

f) Diagrama de Colaboración 44

g) Diagrama de Actividades 45

h) Diagrama de Componentes 46

i) Diagrama de Distribución 48

1.2.2.2 Otras características 49

a) Paquetes 49

b) Notas 50

c) Estereotipos 50

d) Clase Abstracta 51

1.2.2.3 Relaciones 52

a) Asociación 52

b) Agregación 53

c) Generalización 55

1.2.3 Beneficios de la Tecnología a Objetos 56

1.2.4 Servicios Necesarios para implantar la Tecnología 57
Capitulo 2. CONCEPTOS DE BASE DE DATOS 59
2.1 Introducción 59
2.1.1 Cualidades de la Información 60
2.1.2 Concepto de Base de Datos 63
2.1.3 Distintos niveles de abstracción de una Base de Datos 66
2.2 Modelos de Datos 69
2.2.1 Modelos lógicos basados en objetos 69
2.2.1.1. El modelo entidad-relación 69
2.2.1.1.1 Normalizar el Modelo de datos 74
2.2.1.2. El modelo orientado a objetos. 75
2.2.2. Modelos lógicos basados en registros. 76
2.2.2.1. Modelo relacional. 76
2.2.2.1.1 Presentación y Objetivos. 77
2.2.2.1.2 Estructura del Modelo Relacional. 80
2.2.2.2. Modelo de red. 82
2.2.2.2.1 Propuesta Codasyl 83
2.2.2.2.2 Objetivos 83
2.2.2.3. Modelo jerárquico. 84
2.2.2.3.1 Características de la estructura jerárquica 85
2.2.2.4. Diferencias entre modelos. 86
2.2.3. Modelos físicos de datos. 86
2.2.4. Los modelos de datos en el diseño de bases de datos 86
2.3 Seguridad de Bases de Datos 88

Capitulo 3. HERRAMIENTAS DE LIBRE DISTRIBUCIÓN 90
3.1 ¿Qué es el software libre? 90
3.1.1 Como obtener software libre 92
3.1.2 Categorías del software libre y no libre 92
3.2 Sistema operativo LINUX 97
3.2.1 Introducción 97
3.2.2 Historia de Linux 98
3.2.3 ¿Qué son las distribuciones? 100
3.2.4 Características del sistema 101
3.3 Base de datos MySQL 103
3.3.1 ¿Qué es SQL? 103
3.3.2 Historia de MySQL 104
3.3.3 ¿Que es MySQL ? 105
3.3.4 ¿Porque usar MySQL? 105
3.3.5 Principales Características 106
3.4 Lenguajes de Programación 107
3.4.1 historia de PHP 107
3.4.2 Qué es PHP? 108
3.4.3 Qué se puede hacer con PHP? 109
3.5. Servidores Web 110
3.5.1 Apache Web server 110
3.5.2 Estadísticas de servidores Web 112
Capitulo 4. INTERNET 114
4.1 Introducción 114

4.2. Aplicaciones Web 118
4.2.1 La pirámide del diseño Web 118
4.2.2 Construcción de sitios Web 119
4.2.3 Modelo del proceso de desarrollo Web 121
4.2.4 Utilidad y facilidad de empleo 122
4.2.5 ¿Qué es un buen diseño Web? 122
4.2.6 El proceso de diseño Web 124
4.2.6.1 Método Web ad hoc 124
4.2.6.2 Modelo del método Web básico 126
4.2.6.3 Modelo en cascada modificado 127
4.2.6.4 Desarrollo de aplicación conjunta 128
4.2.7 Tipos comunes de sitios Web 129
4.2.7.1 Sitios interactivos frente a sitios estáticos 131
4.2.7.2 Sitios dinámicos 132
4.2.8 Estructuras del sitio 133
4.2.9 Modelos de organización lógica de un sitio 134
4.2.9.1 Lineal 134
4.2.9.1.1 Lineal Básica 135
4.2.9.1.2 Lineal con alternativas 135
4.2.9.1.3 Lineal con opciones 136
4.2.9.1.4 Lineal con caminos laterales 136
4.2.9.2 Rejilla 137
4.2.9.3 Jerarquía 137
4.2.9.3.1 Árboles estrechos 138

4.2.9.3.2 Árboles anchos 138
4.2.9.3.3 Árboles Web 138
4.2.9.3.4 Malla completa 139
4.2.9.3.5 Formas mixtas 139
4.2.9.4 Estilo Web 141
4.2.10 Criterios para la evaluación de Páginas Web 141
4.2.11 El futuro cercano 141
CONCLUSIONES 144
BIBLIOGRAFÍA 150
OTRAS REFERENCIAS 152
GLOSARIO 153
ANEXO 1. Caso Práctico 161
a) Análisis 161
b) Diseño 179
c) Desarrollo 190
d) Implementación 191

"SISTEMA DE SEGUIMIENTO A EGRESADOS DE LA FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN"

INTRODUCCION

La Facultad de Estudios Superiores Cuautitlán cuenta con diversos desarrollos para cubrir sus necesidades de información tanto en el área académica como administrativa.

Estos desarrollos se han hecho de manera aislada por el personal responsable de su respectiva información utilizando diversas plataformas. Conscientes de que no existe una aplicación que controle la información de los alumnos después de haber egresado de la Facultad, lo cual permita tener un seguimiento de los alumnos egresados, se tomó la decisión de llevar acabo dentro de nuestro proyecto de tesis la implementación de un Sistema, con el objetivo de llevar un seguimiento tanto académico, como laboral y profesional de los alumnos egresados, el cual será de gran utilidad a la Licenciatura en informática, así como a las áreas Académicas-Administrativas que así lo requieran.

El sistema empezará a funcionar dentro de la Coordinación de la Licenciatura en informática, pero se espera que pueda serlo para las demás Carreras que se encuentran en la Facultad una vez aprobado.

Con el desarrollo del sistema la Facultad podrá obtener información correcta y en el momento oportuno de los alumnos egresados, permitiendo además de conocer su desarrollo profesional teniendo la posibilidad de seguir contactándolos en caso de requerirlo.

OBJETIVOS

Objetivos generales

•Diseñar y desarrollar un sistema de seguimiento a los alumnos egresados de la Facultad de Estudios Superiores Cuautitlán.

•Manejar información confiable y oportuna de los egresados, que sirva de apoyo en áreas Académicas y Administrativas.

Objetivos particulares

•Emplear herramientas de libre distribución(GNU) [1] para el desarrollo del sistema(Linux, PHP, Mysql, Apache)

•Utilizar la Internet para facilitar el acceso a la información de los egresados.

•Diseñar el modelo de datos del sistema con la metodología UML.

•Realizar el análisis y diseño del sistema orientado a objetos.

ALCANCE DEL SISTEMA

El sistema estará dividido en dos partes que son las siguientes:

Administración:

En esta aparte tendrán acceso únicamente el coordinador de la Licenciatura en Informática, o en dado caso una persona que asigne directamente el coordinador para el mantenimiento de cada uno de los catálogos que requiere el sistema, además de realizar las consultas que tiene el mismo.

[1] Sitio consultado sobre libre distribución es: http://www.gnu.org/philosophy/free-sw.es.html

Egresados:

En esta parte los alumnos egresados de la Licenciatura en Informática, podrán actualizar sus datos académicos, profesionales y laborales entre otros.

Se pueden distinguir los siguientes tipos de usuarios:

- Los que alimentan el sistema de información: será sobre todo los egresados y el coordinador de la Licenciatura en Informática.
- Los que extraen datos del sistema para análisis o procesamiento: será el coordinador de la Licenciatura en Informática.
- **Funciones que proveerá el sistema:**

1. El egresado podrá agregar y actualizar sus datos académicos, labores y profesionales.
2. El egresado podrá dar sugerencias, comentarios o dudas sobre aspectos que le interesan, y serán enviados al administrador para ser respondidos.
3. El egresado podrá consultar avisos que comunique el administrador del sistema.
4. El egresado podrá accesar a los informes que genere el sistema.
5. La coordinación podrá administrar los catálogos que requiere el sistema.
6. La coordinación podrá acceder a las consultas que genere el sistema.

Restricciones que tendrá el sistema:

El sistema inicialmente sólo proveerá servicio directo a los egresados de la Licenciatura en Informática. Sin embargo, el sistema debe proveer la posibilidad de que en caso de requerirse para las demás Carreras pueda ser implementado sin complicaciones.

Es de vital importancia que el sistema sea alimentado por el administrador del sistema, con respecto a los catálogos para empezar a funcionar.

HERRAMIENTAS DE DESARROLLO

Las herramientas utilizadas para el desarrollo del sistema son de libre distribución(GNU), por lo que se pueden obtener sin ningún tipo de costo por lo que el montaje inicial del sistema y su uso no conllevan ninguna inversión. En el Capitulo 3 se detalla más sobre el software de libre distribución.

Hardware

La publicación del sistema será en Internet, utilizando la Red UNAM, y el Servidor donde será hospeda la página será en el Servidor de la Licenciatura en Informática.

Software

Nuestro ambiente de trabajo estará conformado por un servidor de bases de datos, un servidor de web con una extensión llamada PHP.

El sistema estará compuesto por los siguientes elementos:

- Metodología UML para el modelado de datos del sistema

-GNU/Linux como sistema operativo abierto

-Apache como servidor web

-PHP como módulo de ampliación de Apache para acceder a la base de datos

-MySQL como base de datos

-Una interfaz web a MySQL

UML(Lenguaje Unificado de Modelado)

UML no es simplemente un lenguaje para modelamiento orientado a objetos, sino un "lenguaje para modelamiento unificado" relativo a sistemas en general.

El objetivo central del lenguaje es abstraer cualquier tipo de sistema, sea informático o no, mediante diagramas. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibujan de forma que resalten los detalles necesarios para entender el sistema. La manera en que se modela un sistema para ser construido y la elección de los puntos importantes que se reflejan en el modelo junto con los desechados determinan cómo se solucionará el problema y la dificultad de su posterior implementación.

En el capitulo 2 se detallara más sobre el análisis orientado a objetos, así como la metodología UML.

APACHE

Apache es el servidor de web por excelencia. Ha sido uno de los mayores éxitos del software libre y su supremacía entre los servidores de web no se ve amenazada(ni si quiera por el servidor de web de Microsoft)[2].

Su desarrollo continuo y su portabilidad han llevado a plataformas como Windows NT y Windows 95, a parte de casi todos los sistemas Unix.

Una de las principales características de Apache es su extensibilidad basada en una gran modularidad de su código fuente, lo que han facilitado la aparición de módulos de extensión como PHP el cual evitará el uso de cgi-bins por completo, facilitando enormemente la programación de aplicaciones en el lado del servidor, especialmente en el campo de acceso a bases de datos.

PHP: Hypertext Preprocessor (Preprocesador de Hypertexto)

El principal objetivo del lenguaje es permitir a los desarrolladores de aplicaciones basadas en

[2] Para consultar datos estadísticos sobre el servidor apache fue a través del sitio web http://www.netcraft.com

el web escribir páginas que se generan de forma dinámica, sencilla y rápida.

El PHP, desde la versión 3.0 para mantener una cierta asociación con su progenitor, ha sido ampliado con aspectos tan importantes como conceptos de programación orientada a objetos. Lo más espectacular de PHP es su facilidad para acceder a las bases de datos, entre ellas MySql.

MYSQL

MySQL es un sistema de gestión de bases de datos (SGBD) SQL, sus principales objetivos han sido la velocidad y la robustez.

Es un SGBD sencillo y rápido que se adapta perfectamente a entornos en donde el volumen de datos sea del orden de megabytes.

En el mundo de GNU/Linux es MySQL junto a PostgresSQL los principales SGBD de uso libre y con código fuente.

MySQL está al igual que PHP en fase de pleno desarrollo. Se están publicando de forma regular nuevas versiones del sistemas, así como herramientas que son básicas en cualquier SGBD actual.

Se detalla más sobre las herramientas de libre distribución(Linux, MySql, PHP y Apache) que serán utilizadas para el desarrollo del sistema dentro del capitulo 4.

RESULTADOS ESPERADOS

- La información suministrada en todo el sistema deberá ser consistente, correcta, precisa, oportuna, disponible y en un formato que facilite su uso.
- Servirá de apoyo a las áreas Académico-Administrativas.

- Mayor control sobre la información de los alumnos egresados de la Licenciatura en Informática.
- Mejor aprovechamiento de recursos disponibles.
- Una implementación fácil y eficiente con pocas necesidades de Administración.

Se beneficiarían principalmente los que harán uso del sistema, en este caso tanto los egresados como la Facultad de Estudios Superiores Cuautitlán, y los directivos de la Universidad por disponer de un mejor apoyo a sus decisiones.

CAPITULO 1. CONCEPTOS DE ANALISIS Y DISEÑO DE SISTEMAS

1.1 INTRODUCCION

Las metodologías y técnicas estructurales que han reinado en la década de los setenta y ochenta no han eliminado el problema, y de hecho la crisis del software continúa hoy en día. Pese a las muchas herramientas y métodos utilizados, los problemas del diseño descendentes permanecen igual, posiblemente debido a que la complejidad del problema ha crecido considerablemente.

Los cambios realizados en la evolución de un programa son el punto débil de los métodos tradicionales de desarrollo de software, siendo uno de los puntos fuertes de los métodos de desarrollo de software orientado a objetos. Recientemente, las propuestas de reusabilidad o reutilización de componentes de software, se consideran como bloques iniciales para la construcción del programa, de modo similar a la construcción de cualquier objeto complejo que se construye ensamblando sus partes.

Los problemas que se intentan resolver con software implican normalmente elementos de ineludible complejidad, en los que se encuentran una gran cantidad de requisitos, en muchas ocasiones contradictorios. Esta complejidad se produce por las difíciles interacciones entre los usuarios de un sistema y sus desarrolladores: los usuarios encuentran generalmente muy difícil dar precisión sobre sus necesidades de forma que los desarrolladores puedan comprender. En casos extremos, los usuarios pueden tener solo ideas vagas de lo que se desea en un sistema de software.

Por otra parte, los usuarios y desarrolladores tienen diferentes perspectivas de la naturaleza del problema y hacen suposiciones diferentes sobre la naturaleza de la solución. El medio

común de expresar los requisitos hoy día es utilizar un gran volumen de textos, en ocasiones acompañados por esquemas y dibujos. Tales documentos son difíciles de comprender, están abiertos a diferentes interpretaciones y con frecuencia contienen elementos que son diseños en lugar de requisitos esenciales.

Otra complicación frecuente es que los requisitos de un sistema software cambian durante su desarrollo. Esto supone que un sistema grande tiende a evolucionar con el tiempo y el mantenimiento del software en ocasiones es un término que no siempre está bien acuñado.

Para ser más preciso, existen diferentes términos a definir: el mantenimiento busca errores, la evolución responde a cambios de requisitos, y la conservación cuando se utilizan medios para mantener piezas de software en funcionamiento. Desgraciadamente, la realidad sugiere que un porcentaje alto de los recursos de desarrollo de software se gastan en la conservación del software[1].

El tamaño de un programa no es una gran virtud en un sistema de software. Sin embargo la escritura de un gran programa requiere la escritura de grandes cantidades de nuevo software y la reutilización del software existente. Hoy es usual encontrar sistemas en funcionamiento cuyo tamaño se mide en centenares de millares, o incluso millones de líneas de código. Esta característica se facilita descomponiendo nuestra implementación en centenares y a veces millones de módulos independientes. La cantidad de trabajo que conlleva exige el uso de un equipo de desarrolladores, aunque se trata por todos los medios de que este equipo sea lo más pequeño posible. Ahora bien, a medida que haya más desarrolladores, se producen comunicaciones entre ellos, particularmente si el equipo está disperso geográficamente, como suele ser el caso de proyectos grandes.

[1] JOYANES AGUILAR LUIS. Programación Orientada a Objetos segunda edición. Osborne McGraw-Hill, 1998, págs:1-8.

La división de una aplicación en entidades y relaciones que son significativas a los usuarios es un análisis convencional. Con la programación orientada a objetos, este proceso de descomposición se extiende a la fase de implementación. Es más fácil diseñar e implementar aplicaciones orientadas a objetos, ya que los objetos en el dominio de la aplicación se corresponden directamente con los objetos en el dominio del software.

El software ofrece flexibilidad, de modo que es posible para un desarrollador expresar prácticamente cualquier clase de abstracción.

Los sistemas orientados a objetos proporcionan el rendimiento, la flexibilidad y funcionalidad requerida para implementaciones prácticas. La programación orientada a objetos expande también la variedad de aplicaciones que se pueden programar, debido a que se liberan las restricciones de los tipos de datos predefinidos y acomoda estructuras de datos heterogéneas y complejas. Se pueden añadir nuevos tipos de datos y modificar código existente.

Por consiguiente, la ***Programación Orientada a Objetos(POO)*** no sólo son nuevos lenguajes de programación, sino un nuevo modo de pensar y diseñar aplicaciones que pueden ayudar a resolver problemas que afectan al desarrollo de software. Sin embargo, el lenguaje debe ser capaz de soportar el nuevo paradigma, siendo una parte esencial de esta revolución.

1.1.1 BREVE REVISION HISTORICA

Empezaremos por hacer una revisión de la evolución de los lenguajes de programación, y continuaremos explicando qué es la POO para terminar haciendo una breve revisión histórica de la misma.

1.1.1.1 EVOLUCION DE LOS LENGUAJES DE PROGRAMACION

Toda la historia de los lenguajes de programación se han desarrollado en base a una sola idea conductora: hacer que la tarea de realizar en la computadora sea cada vez lo más simple, flexible y portable posible.

La POO supone, no solo un nuevo paso hacia ese fin. A medida que se van desarrollando los lenguajes, se va desarrollando también la posibilidad de resolver problemas cada vez más complejos. En la evolución de cada lenguaje, llega un momento en el que los programadores comienzan a tener dificultades a la hora de manejar programas que sean de un cierto tamaño y sofisticación.

Esta evolución en los lenguajes, ha venido impulsada por tres motores bien distintos:

Los avances tecnológicos

Los avances conceptuales

Los avances en cuanto a enfoque de la programación

·**Evolución en cuanto a la tecnología**

En un principio las computadoras no eran más que un conjunto de microinterruptores, es decir eran relés, que no es más que un electroimán que puede cerrar o abrir un circuito eléctrico.

Un programa de computadora no es más que una sucesión de instrucciones que se ejecutarán secuencialmente, es decir, una detrás de otra. Por tanto, como la forma de programarlos es introduciendo secuencias de ceros y unos; puede imaginarse la gran probabilidad que existe de error al introducir sucesiones enormemente largas; además, una vez cometido un error encontrarlo y corregirlo es muy difícil de hacerlo.

Cronológicamente el primer avance importante vino con la aparición de los lenguajes

ensambladores. A estos lenguajes los llamamos de bajo nivel ya que se hallan estrechamente ligados a la forma de trabajo de la máquina con la que programamos. El problema de los lenguajes ensambladores es que se hallan fuertemente comprometidos con la máquina. Un programa escrito en lenguaje ensamblador solo podrá ser ejecutado en el tipo de máquina donde fue diseñado.

El siguiente paso, vino con la aparición de los lenguajes de alto nivel. El proceso de desarrollo de una aplicación con un lenguaje de alto nivel es mucho más rápido, simple y, por tanto, resulta más fácil detectar y corregir errores. Esto se debe principalmente a dos factores: por un lado, cada instrucción que escribimos en lenguaje de alto nivel puede equivaler a varias decenas e incluso cientos de instrucciones en ensamblador; por otra parte, la sintaxis de las instrucciones y los nemónicos que usamos se parecen algo más al lenguaje cotidiano.

Realizar un programa que en un lenguaje de alto nivel nos puede llevar pocos días, en ensamblador es tarea de meses. Si tenemos esto en cuenta, comprenderemos que después de escribir un programa en código máquina y de haberlo depurado para eliminar los errores, a nadie le queden ganas de ponerse a re-escribirlo para otra máquina distinta. Con lo que la portabilidad queda mermada considerablemente, hecho que no debe ocurrir con un lenguaje de alto nivel.

La cuestión tiene dos vertientes bien distintas: por un lado, los programas realizados con lenguajes de alto nivel, realizan los mismos procesos de un modo más lento que de haberlos escrito con uno de bajo nivel, este problema no es demasiado gravoso si echamos un vistazo a la evolución en la potencia de cálculo de las máquinas en los últimos diez años. Salvo para programas muy específicos, o de gran rendimiento, no habria ningún problema, en escribir nuestras aplicaciones en un lenguaje de alto o bajo nivel.

En teoría, un lenguaje de alto nivel es independiente de la máquina donde se ejecute, el mismo código que escribió para una máquina puede ser traducido al lenguaje hexadecimal en esa máquina o en otra cualquiera que disponga de traductor para ese lenguaje.

·Evolución en cuanto a la conceptualización

El primer avance en metodología de programación, vino con la Programación Estructurada, en este concepto vamos a incluir el propio y el de técnicas de Programación con Funciones, también llamado Procedural, ya que ambos se hallan íntimamente relacionados, no creemos, que se pueda concebir la programación estructurada sin el uso masivo de funciones.

La programación en ensamblador es lineal, es decir, las instrucciones se ejecutan en el mismo orden en que las escribimos. Podemos, sin embargo, alterar este orden haciendo saltos desde una instrucción a otro lugar del programa distinto a la instrucción que le sigue a la que se esta procesando. Este sistema de trabajo es complicado, ya que obliga al programador a retener en su cabeza permanentemente todo el código escrito hasta un momento determinado para poder seguir escribiendo el programa; además a la hora de leerlo, el programador se pierde con facilidad porque debe ir saltando continuamente de unos trozos de código a otros.

·Programación Lineal

Cada línea de programa debe ir precedida de un identificador para poder referenciarla.

·Programación Estructurada

Cuando escribimos un programa usando técnicas de programación estructurada, los saltos no se recomienda utilizarlos, por no decir prohibidos, lo que no es nada conveniente si queremos entender algo que escribimos hace tres meses de forma rápida y clara.

De hecho, cuando el traductor (ya sea intérprete o compilador) cambia nuestro programa a

código máquina, lo convierte a estilo lineal, pero eso es asunto de la computadora, nosotros escribimos y corregimos nuestro programa de modo claro, y podemos seguir el flujo de la información con facilidad. El programador puede hacer programas cada vez más extensos sin perderse en un entramado de líneas de código interdependientes. Dentro de la programación estructurada aparece un concepto que nos permite abarcar programas más amplios con menor esfuerzo: el de función esto se refiere a que muchas veces realizo procesos que se repiten y en los que sólo cambia algún factor, si trato ese proceso como un subprograma al que llamo cada vez que lo necesito, y cada vez que lo llamo puedo cambiar ese factor, estaré reduciendo el margen de error, al reducir el número de líneas que necesito en mi programa, ya que no tengo que repetir todas esas líneas cada vez que quiera realizar el proceso, con una sola línea de llamada al subprograma será suficiente; además, de haber algún fallo en este proceso el error queda circunscrito al trozo de código de la función.

- ***Programación Orientada a Objetos***

Hasta ahora, el control del programa era tarea del programador, si usted ha realizado algún programa de magnitud considerable lo habrá padecido. El programador tenía que controlar y mantener en su mente cada proceso que se realizaba y los efectos colaterales que pudieran surgir entre distintos procesos, lo que llamamos colisiones. En POO, el programa se controla a si mismo y la mente del programador se libera enormemente pudiendo realizar aplicaciones mucho más complejas al exigir menor esfuerzo de atención, ya que los objetos son entidades autónomas que se controlan a sí mismos. Esto es posible principalmente porque los objetos nos impiden mezclar sus datos con otros métodos distintos a los suyos.

En programación estructurada, una función trabaja sobre unos datos, y no debería modificar datos que no le corresponden hacer, pero de eso tiene que encargarse el programador, en POO es el propio sistema de trabajo el que impide que esto ocurra. Además, la re-usabilidad

del código escrito es mucho mayor que con el uso de funciones, y la portabilidad es también mayor.

-Evolución en cuanto al enfoque

La evolución de los lenguajes de programación, en cuanto a enfoque es también una evolución conceptual, pero ésta es tan profunda que supone un cambio drástico en cuanto al modo de concebir el tratamiento de la programación.

Existen dos o tres enfoques diferentes[2]:

- ***Programación Procedural***

En ellos, debemos establecer, hechos (datos), reglas para el manejo de esos datos y de decisión y tenemos que decirle al lenguaje como alcanzar el objetivo que se persigue. Es decir, donde buscar la información, como manipularla, cuando parar, etc. Algunos lenguajes son los siguientes Java, C, Pascal, Basic, Cobol, Fortran, APL, RPG, Clipper, etc.

- ***Programación Declarativa***

Se basan en manipulaciones lógicas, utiliza la lógica proposicional -lógica de predicados- para realizar deducciones. Solo se declaran hechos, es la maquinaria del lenguaje quien se encargará de extraer las conclusiones que resulten inferibles de estos hechos.

A esta maquinaria se le llama motor de inferencias, que es, por otro lado, el corazón de un Sistema Experto. Uno de los lenguaje más conocidos que existen hasta ahora es Prolog.

- ***Programación Orientada a Objetos***

La POO es un conjunto de técnicas, donde una técnica de programación no es,

[2] La clasificación de la programación fue extraída de:
MEYER BERTRAND.:Object Oriented Software Construcción(OOSC) segunda edicion. Englewood, Cliffs NJ. Prentuce-Hall. 1997.

obviamente, un lenguaje, pero puede aplicarse a cualquier lenguaje. Podemos definirlo como un conjunto de reglas a seguir para hacernos la tarea de programar más fácil. Son consejos de expertos programadores, que tras años de trabajo, han acumulado una gran experiencia. Pero estas técnicas, son independientes del lenguaje en que trabajemos. Un lenguaje de programación, es un conjunto de instrucciones entendibles directamente o traducibles al lenguaje del ordenador con el que trabajemos combinando estas instrucciones realizamos programas.

1.1.2 ¿QUÉ ES LA POO?

La POO no es un lenguaje. De hecho las técnicas de POO pueden utilizarse en cualquier lenguaje conocido y los que están por venir, aunque estos últimos, al menos en los próximos años, incluirán facilidades para el manejo de objetos. Desde luego, que en los lenguaje que prevén el uso de objetos la implementación de técnicas de POO resulta mucho más fácil y provechosa que los otros. Se pueden utilizar estos lenguajes sin que los programas resultantes tengan nada que ver con la POO.

Esta es un conjunto de técnicas que nos permiten incrementar enormemente nuestro proceso de producción de software; aumentando drásticamente nuestra productividad por un lado y permitiéndonos abordar proyectos de mucha mayor envergadura por otro.

Usando estas técnicas nos aseguramos la re-usabilidad de nuestro código, es decir, los objetos que hoy escribimos, si están bien escritos nos servirán para siempre. Podemos re-usar ciertos comportamientos de un objeto, ocultando aquellos otros que no nos sirven, o redefinirlos para que los objetos se comporten de acuerdo a las nuevas necesidades.

En la programación tradicional, nos hubiésemos visto obligados a construir un coche nuevo

por completo. En POO si queremos construir un objeto que comparte ciertas cualidades con otro que ya tenemos creado, no tenemos que volver a crearlo desde el principio; simplemente, decimos que queremos usar del antiguo en el nuevo y que nuevas características tiene nuestro nuevo objeto, aún más podemos incorporar objetos que otros programadores han construido en nuestros programas. Además de modificar comportamientos de los objetos construidos por otros programadores sin tener que saber como los han construido ellos.

1.1.3 PROGRAMACION ORIENTADA A OBJETOS

La programación orientada a objetos(POO) se suele conocer como un nuevo paradigma de programación. La Orientación a objetos es especialmente adecuada para realizar determinadas aplicaciones, sobre todo realización de prototipos y simulación de programas.

Los mecanismos de encapsulación de POO soportan un alto grado de reutilización de código, que se incrementa por sus mecanismos de herencia. En el entorno de las bases de datos, la POO se adjunta bien a los modelos semánticos de datos para solucionar las limitaciones de los modelos tradicionales, incluso el modelo relacional.

Aumento de Lenguajes de Programación Orientada a Objetos. Interfaces de usuario gráficos y visuales. Las interfaces de usuario de una aplicación manipulan la entrada y salida del usuario. Por consiguiente, su función principal es la comunicación con el usuario final; la entrada al sistema se puede controlar a través de líneas de órdenes o alternativamente el usuario puede interactuar con el sistema, con construcciones de programación visuales.

1.1.3.1 CONCEPTOS BASICOS

La orientación a objetos puede describirse como el conjunto de disciplinas que desarrollan[3] y

[3] JOYANES AGUILAR LUIS.:Programación Orientada a Objetos segunda edición, Osborne McGraw-Hill, 1998, págs: 18-24.

modelan software que facilitan la construcción de sistemas complejos a partir de componentes. La programación orientada a objetos permite una representación más directa del modelo del mundo real en el código. El resultado es la transformación normalmente llevada a cabo de los requisitos del sistema a la especificación del sistema se reduce considerablemente.

Los conceptos y herramientas orientados a objetos son tecnologías que permiten que los problemas del mundo real sean expresados de modo fácil y natural. Las técnicas orientadas a objetos proporcionan mejoras y metodologías para construir sistemas de software complejos a partir de unidades de software modularizado y reutilizable. Se necesita un nuevo enfoque para construir software en la actualidad. Este nuevo enfoque debe ser capaz de manipular tanto sistemas grandes como pequeños y debe crear sistemas fiables que sean flexibles, mantenibles y capaces de evolucionar para cumplir las necesidades de cambio.

La tecnología orientada a objetos puede cubrir estos cambios y algunos otros más en el futuro. Además trata de cumplir las necesidades de los usuarios finales, así como las propias de los desarrolladores de software. Estas tareas se realizan mediante la modelación del mundo real, el soporte fundamental es el modelo objeto.

a) Definición de Clase

Una clase, es simplemente una abstracción que hacemos de nuestra experiencia sensible. El ser humano tiende a agrupar seres o cosas(objetos) con características similares en grupos(clases). Cada clase posee unas cualidades que diferencian de las otras. Por ejemplo, los vegetales se diferencian de los animales y los minerales, en que los primeros son seres vivos y los minerales no. De los animales se diferencian en que las plantas son capaces de sintetizar clorofila a partir de la luz solar y los animales no.

b) Definición de Objeto

En POO, un objeto es un conjunto de datos y métodos. Los datos son lo que antes hemos llamado características o atributos, los métodos son los comportamientos que pueden realizar.

Lo importante de un sistema POO es que ambos, datos y métodos están tan intrínsicamente ligados que forman una misma unidad conceptual y operacional. En POO, no se pueden desligar los datos de los métodos de un objeto. Así es como ocurre en el mundo real.

Una clase es un nuevo tipo de dato y un objeto cada una de las asignaciones que hacemos a ese tipo de dato[4].

Los cuatro elementos más importantes de este modelo son:

Abstracción, Encapsulación, Polimorfismo, Herencia

c) Abstracción

La abstracción es la capacidad para encapsular y aislar la información del diseño y ejecución. En otro sentido, las técnicas orientadas a objetos se ven como resultado de una larga progresión histórica que comienza en los procedimientos y sigue en los módulos, tipos abstractos de datos y objetos.

La abstracción es la clave para diseñar buen software. En los primeros días de la informática, los programadores enviaban instrucciones binarias a una computadora, manipulando directamente interrupciones en sus paneles frontales.

Secuencias de sentencias de lenguajes de alto nivel se pueden agrupar en procedimientos y se invocan por una sentencia. La programación estructurada alienta el uso de abstracciones de control, tales como bucles o sentencias if-then, que se han incorporado en lenguajes de

[4] Conceptos y elementos del modelo orientado a objetos fueron extraídos de:
BOOCH, GRADY.: Análisis y Diseño Orientado a Objetos con Aplicaciones, Addison-Wesley Diaz de Santos, 1995

alto nivel. Estas sentencias de control permitieron a los programadores abstraer las condiciones comunes para cambiar la secuencia de ejecución.

El proceso de abstracción fue evolucionado desde la aparición de los primeros lenguajes de programación. El método más idóneo para controlar la complejidad fue aumentar los niveles de abstracción. En esencia, la abstracción supone la capacidad de encapsular y aislar la información del diseño y ejecución. En un determinado sentido, las técnicas orientadas a objetos pueden verse como un producto natural de una larga progresión histórica, que va desde las estructuras de control, pasando por los procedimientos, los módulos, los tipos abstractos de datos y los objetos.

d) Encapsulación

La encapsulación es la propiedad que permite asegurar que el contenido de la información de un objeto está oculta al mundo exterior. También se le conoce como ocultación de información, es el proceso de ocultar todos los secretos de un objeto que no contribuyen a sus características esenciales.

El encapsulamiento (o encapsulación) es cuando un objeto trae consigo su funcionalidad, esta última se oculta. El encapsulamiento permite reducir el potencial de errores que pudieran ocurrir. En un sistema que consta de objetos, estos dependen unos de otros en diversas formas. Si uno de ellos falla y los especialistas de software tiene que modificarlo de alguna forma, el ocultar sus operaciones de otros objetos significará que tal vez no será necesario modificar los demás objetos.

En el mundo real, también se ve la importancia del encapsulamiento en los objetos con los que trabaje. Por ejemplo, el monitor de su computadora, en cierto sentido, oculta sus operaciones de la CPU, es decir, si algo falla en su monitor, lo repara o lo reemplazará pero es muy probable que no tenga que reparar o reemplazar la CPU al mismo tiempo que el

monitor.

e) Polimorfismo

Por polimorfismo entendemos aquella cualidad que poseen los objetos para responder de distinto modo ante el mismo mensaje.

Pongamos por ejemplo las clases *hombre*, *vaca* y *perro*, si a todos les damos la orden - enviamos el mensaje- Come, cada uno de ellos sabe cómo hacerlo y realizará este comportamiento a su modo. Realmente, y para ser exactos, los mensaje no se envían a las clases, sino a todos o algunos de los objetos instanciados de las clases.

En POO el mismo nombre nos sirve para todas las clases creadas si así lo queremos, lo que suele ser habitual. El mismo nombre suele usarse para realizar acciones similares en clases diferentes. El polimorfismo nos facilita el trabajo, ya que gracias a él, el número de nombres de métodos que tenemos que recordar disminuye.

La mayor ventaja la obtendremos en métodos con igual nombre aplicados a las clases que se encuentran próximas a la raíz del árbol de clases, ya que estos métodos afectarán a todas las clases que de ellas se deriven.

f) Herencia

Esta es la cualidad más importante de un sistema POO. Cada vez que creamos una clase heredada de otra (la padre), añadimos métodos a la clase padre o modificamos alguno de los métodos de la clase padre.

Las cualidades comunes que comparten distintas clases, pueden y deben agruparse para formar una clase padre también llamada superclase; esta poseería los métodos comunes a todas ellas y sólo tendríamos que añadir aquellos métodos propios de cada una de las subclases, pudiendo reutilizar el código escrito en la superclase desde cada una de las clases derivadas.

La herencia como puede intuir, es la cualidad más importante de la POO, ya que le permite reutilizar todo el código escrito para las superclases re-escribiendo solo aquellas diferencias que existan entre éstas y las subclases. A la clase heredada se le llama, *subclase* o *clase hija*, y a la clase de la que se hereda *superclase* o *clase padre*.

Al heredar, la clase heredada toma directamente el comportamiento de su superclase, pero puesto que ésta puede derivar de otra, y esta de otra, etc., una clase toma indirectamente el comportamiento de todas las clases de la rama del árbol de la jerarquía de clases a la que pertenece.

Se heredan los datos y los métodos, por lo tanto, ambos pueden ser redefinidos en las clases hijas, aunque lo más común es redefinir métodos y no datos. Una subclase hereda de su superclase sólo aquellos miembros visibles desde la clase hija y por lo tanto solo puede redefinir estos.

Una subclase tiene forzosamente que redefinir aquellos métodos que han sido definidos como abstractos en la clase padre o padres. Se redefinen los métodos, aun cuando a veces se hace necesario redefinir datos de las clases superiores. Al redefinir un método queremos o bien sustituir el funcionamiento del método de la clase padre o bien ampliarlo.

En el primer caso (sustituirlo) no hay ningún problema, ya que a la clase hija la dotamos con un método de igual nombre que el método que queremos redefinir en la clase padre y lo implementamos según las necesidades de la clase hija. De este modo cada vez que se invoque este método de la clase hija se ejecutará su código, y no el código escrito para el método homónimo de la clase padre.

Pero si lo que queremos es ampliar el funcionamiento de un método existente en la clase padre, entonces primero tiene que ejecutarse el método de la clase padre, y después el de la clase hija. Pero como los dos métodos tienen el mismo nombre, se hace necesario habilitar

alguna forma de distinguir cuando nos estamos refiriendo a un método de la clase hija y cuando al del mismo nombre de la clase padre.

Esto se hace mediante el uso de dos palabras reservadas, las cuales pueden variar dependiendo del lenguaje que se utilice, pero que normalmente son: **this** (en algunos lenguajes se utiliza la palabra reservada *Self*) y super.

this

Con esta palabra, podemos referirnos a los miembros de la clase. De hecho, siempre que dentro del cuerpo de un método nos refiramos a cualquier miembro de la clase, ya sea una variable u otro método, podemos anteponer *this*, aunque en caso de no existir duplicidad, el compilador asume que nos referimos a un miembro de la clase. Al utilizar *this* para dejar claro que se está haciendo referencia a un miembro de la clase.

super

Al contrario que *this*, *super* permite hacer referencia a miembros de la clase padre (o a los ancestros anteriores, que no hayan sido ocultados por la clase padre) que se hayan redefinido en la clase hija. Si un método de una clase hija redefine un miembro -ya sea variable o método- de su clase padre, es posible hacer referencia al miembro redefinido anteponiendo *super*.

Existen dos mecanismos de herencia utilizados comúnmente en programación orientada a objetos: herencia simple y herencia múltiple.

- ***Herencia Simple***

Un objeto(clase) puede tener sólo un ascendiente, o dicho de otro modo, una subclase puede heredar datos y métodos de una única clase, así como añadir o quitar comportamientos de la clase padre.

·Herencia Múltiple

Es la propiedad de una clase de poder tener más de un ascendiente inmediato, o lo que es igual, adquirir datos y métodos de más de una clase. Prácticamente todo lo que se hace con herencia múltiple se puede hacer con herencia simple, aunque resulta a veces más difícil. Una dificultad surge con la herencia múltiple cuando se combinan diferentes tipos de objetos, cada uno de los cuales define métodos o campos iguales.

Por ejemplo se tiene las clase Gráficos y Sonidos, y se crea un nuevo objeto denominado Multimedia a partir de ellos. Gráficos tiene tres campos datos: tamaño, color y mapa de bits, y los métodos dibujar, cargar y escala; Sonidos tiene dos campos de datos duración, voz y los métodos reproducir, cargar, escala. Así, para un objeto Multimedia, el método escala significa poner el sonido en diferentes tonalidades, o bien aumentar/reducir el tamaño de escala del Gráfico.

El problema que se produce es la ambigüedad, y se tendrá que resolver con una operación de prioridad que el correspondiente lenguaje deberá soportar y entender en cada caso.

En realidad, ni la herencia simple ni la herencia múltiple son perfectas en todos los casos, y ambas pueden requerir un poco más de código extra que represente bien las diferencias en el modo de trabajo.

g) Mensaje

Un mensaje es una petición de un objeto a otro objeto al que le solicita ejecutar uno de sus métodos. Un mensaje incluye el nombre de una operación y cualquier argumento requerido por esa operación.

Los objetos se comunican entre sí a través del uso de mensajes. Al enviar un mensaje, se ejecuta un método, el cual puede llamar a su vez a otros métodos de su clase o de cualquier otra clase o bien cambiar los valores de los datos de ese objeto. Si el programador tiene que

alterar los valores de los datos de un objeto deberá mandar un mensaje a ese objeto; lo mismo sucede cuando un objeto tiene que cambiar los valores de los datos de otro objeto. Como podemos apreciar, un objeto es como una caja negra, a la que se le envía un mensaje y éste responde ejecutando el método apropiado, el cual producirá las acciones deseadas. un objeto, una vez programado es solo manipulable a través de mensajes.

h) Sobrecarga

La sobrecarga es una propiedad que describe una característica adecuada que utiliza el mismo nombre de operación para representar operaciones similares que se comportan de modo diferente cuando se aplican a clases diferentes. Por consiguiente, los nombres de las operaciones se pueden sobrecargar, esto es, las operaciones se definen en clases diferentes y pueden tener nombres idénticos aunque su código programado puede diferir.

Si los nombres de una operación se utilizan para nuevas definiciones en clases de una jerarquía, la operación a nivel inferior se dice que anula la operación aun nivel más alto.

Actualmente la sobrecarga se aplica sólo a operaciones. La sobrecarga no es una propiedad específica de los lenguajes orientados a objetos, ya que los lenguajes de programación convencionales soportan sobrecarga para algunas operaciones sobre algunos tipos de datos.

1.1.4 CARACTERISTICAS DE LOS LENGUAJES ORIENTADOS A OBJETOS

Tipificación estricta(fuerte). Es el proceso de declarar el tipo de información que puede contener una variable. Los errores de programación relacionados con el número de parámetros, tipos de parámetros e interfaces de módulos, se detectan durante las fases de diseño e implementación, en lugar de en tiempos de ejecución.

Encapsulamiento. Es deseable que el lenguaje soporte ocultación de la información, mediante partes independientes, para la especificación y la implementación. Esta característica proporciona un diseño débilmente acoplado que cumple con rigor el principio básico de la inferencia de software: acoplamiento débil y fuerte cohesión entre los módulos de un programa.

Generalidad. Las clases parametrizadas sirven para soportar un alto grado de reusabilidad. Estos elementos genéricos se diseñan con parámetros formales, que se instanciarán con parámetros reales, para crear instancias de módulos que se compilan, enlazan y ejecutan posteriormente.

Pasos de mensajes. El lenguaje es conveniente que soporte paso bidimensional de mensajes entre módulos, lo que implicará módulos débilmente acoplados y diseños flexibles. Esto significa que se deben poder pasar señales entre módulos, sin necesidad de tener que pasar realmente ningún dato.

Polimorfismo. Los lenguajes deben permitir que existan operaciones con igual nombre, que se utilicen para manejar objetos de tipos diferentes en tiempo de ejecución. El polimorfismo se implementa, normalmente, en unión con la herencia.

Excepciones. Se debe poder detectar, informar y manejar condiciones de excepciones utilizando construcciones de lenguaje. Esta propiedad añadida al soporte de tolerancia a fallos de software permitirá una estrategia de diseño eficiente.

Concurrencia. Es conveniente que el lenguaje soporte la creación de procesos paralelos independientes del sistema operativo. Esta propiedad simplificará la transportabilidad de un sistema de tiempo real de una plataforma a otra.

Persistencia. Los objetos deben poder ser persistentes, es decir, los objetos han de poder permanecer después de la ejecución del programa.

Datos compartidos. Los módulos se deben poder comunicar mediante memoria compartida, además del paso de mensajes.

Los lenguajes de programación disponibles actualmente no cumplen todas las características citadas anteriormente. En general, los lenguajes orientados a objetos no soportan concurrencia.

1.1.5 BENEFICIOS DE LAS TECNOLOGÍAS DE OBJETOS

Una de la principal razón, es la fiabilidad y productividad del desarrollador. La fiabilidad se puede mejorar, debido a que cada objeto es simplemente <<una caja negra>>con respecto a objetos externos con los que debe comunicarse. Las estructuras de datos internos y métodos se pueden refinar sin afectar a otras partes de un sistema.

Los sistemas tradicionales, por otra parte, presentan con frecuencia efectos laterales no deseados. Las tecnologías de objetos ayudan a los desarrolladores a tratar la complejidad en el desarrollo del sistema. La productividad del desarrollador se puede mejorar, debido a que las clases de objetos se pueden hacer reutilizables de modo que cada subclase o instancia de un objeto puede utilizar el mismo código de programa para la clase. Por otra parte, esta productividad también aumenta, debido a que existe una asociación más natural entre objetos del sistema y objetos del mundo real.

1.1.6 LIMITACIONES E INCONVENIENTES DE LA POO

La POO, como todo en este mundo, también tiene sus limitaciones e inconvenientes. Desde la perspectiva de la POO, conlleva un análisis mucho más riguroso y tedioso. Sin embargo, que lo vemos como un inconveniente, también, una vez realizado nos permitirá implementar

nuestra aplicación en un tiempo mucho menor.

El mayor inconveniente real proviene de un "error" de planteamiento; y como casi siempre, estos son de mucha mayor dificultad a la hora de solucionarlos. El problema, según comenta uno de los mejores analistas de hoy en día, Jeff Dunteman, es que "La encapsulación y la herencia se hallan en esquinas opuestas de la casa". Es decir, la encapsulación choca frontalmente con la herencia, y sin embargo, son dos piedras angulares de la POO.

Por un lado decimos que los objetos deben ser totalmente independientes y autónomos, y por otro, al heredar unas clases de otras, estamos dejando fuera de un objeto perteneciente a una clase hija gran parte de la información que éste necesita para poder comportarse.

Otro problema más inmediato, más real y menos teórico: el que estriba en la imposibilidad de utilización conjunta de objetos de distintos programadores. Veamos un ejemplo simple, supongamos que estoy montando un coche en mi garaje, puedo comprar un carburador Fiat y montarlo en un coche Opel, unos asientos Volvo y montarlo en una carrocería Citroën; esto mismo no puedo hacerlo en POO.

Si Microsoft fabrica la clase *Menu*, que deriva de la clase *Visual* y Borland fabrica la clase *Ventana* aunque también derive de la clase *Visual*, yo no puedo tomar el menú de una y la ventana de la otra, a menos que todas las clases superiores (en este caso solo una) sean exactamente iguales: tengan el mismo nombre, contengan los mismos datos y los mismos métodos y en estos, todos los parámetros deben coincidir en orden y en tipo.

Este problema por ahora no tiene solución, y lo peor es que no se vislumbra que la tenga en un futuro próximo, ya que para ello sería necesario normalizar las clases (al menos las más habituales).

RESUMEN:

Después de haber explicado los elementos que contiene la programación orientada a objetos, hacemos mención que la programación utilizada para el desarrollo del sistema fue a través de la orientada a objetos, además de utilizar la metodología UML.

Donde cabe aclarar que aunque UML es orientado a objetos preferentemente, es útil en cualquier modelo tecnológico ya que es independiente de lenguajes de programación o tecnología determinada. En el siguiente tema contiene definiciones, características, elementos de la Metodología UML, que se utilizaron para realizar el modelado de datos del sistema.

1.2 UML (LENGUAJE UNIFICADO DE MODELADO)

1.2.1 INTRODUCCION

El Lenguaje Unificado de Modelado(en ingles Unified Modeling Languaje) es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, conocidos recientemente como "los tres amigos", trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa. Cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los años noventas empezaron a intercambiar ideas entre sí y decidieron desarrollar su trabajo en conjunto.

En 1994 Rumbaugh ingreso a Rational Software Corporation, donde ya trabajaba Booch. Jacobson ingresó a Rational un año después.

UML unifica por un lado Grady Booch, autor del método Booch; James Rumbaugh, autor de OMT(Object Modeling Technique) e Ivar Jacobson autor OOSE(Object Oriented Software Engineering), pero se espera que sea algo más que eso. UML[1] está sufriendo un proceso de estandarización por el grupo OMG, además del comité ISO/ANSI.

Los anteproyectos del UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio del UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de administración de objetos) como respuesta a su propuesta

[1] Para detalles más completos y actualizaciones de UML consultar el sitio web de Rational en http://www.rational.com

para un lenguaje de modelado estándar.

El consorcio aumentó y generó la versión 1.1 misma que se puso nuevamente a consideración del OMG. El grupo adoptó esta versión a finales de 1997. El OMG se encargó de la conservación del UML y produjo otras versiones en 1998. El UML ha llegado a ser el estándar de modelado en la industria del software, y su evolución continúa.

1.2.2 ELEMENTOS DE UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. En lugar de indicarle a usted cuales son los elementos y las reglas[2].

1.2.2.1 DIAGRAMAS DE UML

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se le conoce como Modelo. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

a) Diagrama de Clases

Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares.

Un diagrama de clases es un tipo de modelo, especificamente un tipo de modelo estático. Un diagrama de clase describe la vista estática de un sistema en términos de clases y relaciones entre ellos. Aunque tiene semejanzas con el modelo de datos, recordemos que las clases no

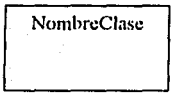
[2] Conceptos y Notación de símbolos consultados en:
SCHMULLER JOSEPH.: Aprendiendo UML en 24 horas, Pearson Education, 1999.

sólo muestran las estructuras de nuestra información sino que describen también su comportamiento.

Una clase es representada por un rectángulo, por convención una palabra con la primera letra en mayúscula y se coloca en la parte superior del rectángulo, es conveniente que el nombre se deduzca del dominio del problema y sea lo menos ambiguo posible.

Esta se divide en tres áreas. El área superior contiene el nombre, el área central contiene los atributos, y el área inferior las acciones.

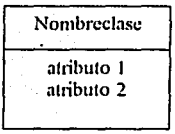
La representación de una clase.



Atributos

Un atributo es una propiedad o característica de una clase y describe un rango de valores que la propiedad podrá contener en los objetos. Una clase podrá contener varios o ningún atributo. Por convención, el atributo se escribe en minúsculas.

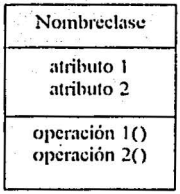
Representación de una clase y sus atributos



Operaciones

Una operación es algo que la clase puede realizar, o que usted (u otra clase) pueden hacer a una clase. De la misma manera que el nombre de los atributos el nombre de una operación se escribe en minúsculas.

Representación de una clase con sus operaciones



Visibilidad de atributos y operaciones

Los atributos pueden tener diferente visibilidad. La visibilidad describe si el atributo es visible y puede ser referenciado por otras clases distintas de aquellas en que se ha definido.

Los atributos y su representación grafica pueden ser:

Públicos: Se pueden utilizar y visualizar fuera de la clase (se representa con un signo más delante del nombre).

Privados: No se puede acceder al mismo desde otra clase (se representa con un signo menos delante del nombre).

Protegidos: Se utiliza en notaciones de generalización y especificación para representar atributos protegidos relacionados con clases derivadas.

La visibilidad de los atributos usuales son públicos y privados, si ningún signo se visualiza significa que la visibilidad no está definida.

La sintaxis formal de un atributo es:

visibilidad nombre: tipo-expresión = valor_inicial{cadena-propiedad}

Ejemplo de visibilidad de atributos

Factura
+ cantidad: Real + cliente: String - administrador: String = "director" - numero de factura: Integer

Las operaciones de una clase se utilizan para manipular los atributos (modificar el estado de la clase) o realizar alguna otra acción; son funciones a las que se llama o invoca, son interiores a la clase y se pueden aplicar sobre objetos de esa clase. La signatura de la operación describe todo lo que necesita utilizar la operación; contiene el tipo de retorno, nombre y parámetros.

Las operaciones describen lo que puede hacer la clase, es decir, sus servicios. Al igual que los atributos, las operaciones también tienen su visibilidad (+ pública, - privada, # protegida) .

La sintaxis formal de una operación es:

visibilidad nombre (lista-parámetros): tipo-expresión-retorno

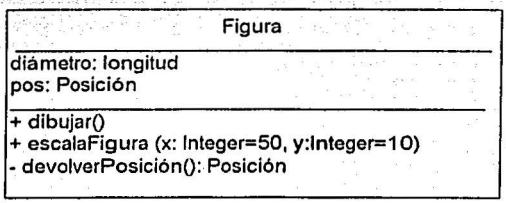
{cadena-propiedad}

Donde lista-parámetros es una lista de parámetros formales separados por coma, especificados cada uno con la siguiente sintaxis:

nombre: tipo-expresión = valor por omisión

No es necesario que todas las operaciones tengan tipo de retorno, parámetros o cadena-prioridad, pero siempre han de tener una signatura única (= tipo retorno, nombre, parámetros).

Ejemplo de la Clase Figura con atributos y operaciones



Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí.

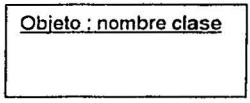
Décadas de experiencia sugieren que es más sencillo desarrollar aplicaciones que simulen algún aspecto del mundo cuando el software representa clases de cosas reales. Los diagramas facilitan las representaciones a partir de las cuales los desarrolladores podrán trabajar. A su vez, los diagramas de clases colaboran en lo referente al análisis. Permiten al analista hablarle a los clientes en su propia terminología, lo cual hace posible que los clientes indiquen importantes detalles de los problemas que requieren ser resueltos.

b) Diagrama de Objetos

Un objeto es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones). El nombre de la instancia se encuentra a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha.

Un objeto de una clase se representa por un rectángulo con el nombre del objeto y su clase subrayada.

Ejemplo de notación de un objeto



c) Diagrama de Casos de Uso

Este tipo de diagrama es particularmente crucial para la fase de análisis del desarrollo de un sistema. La forma en que los usuarios utilicen un sistema le da la pauta para lo que diseñará y creará. El caso de uso es una estructura que ayuda a los analistas a trabajar con los usuarios para determinar la forma en que se usará un sistema. Con una colección de casos de uso se puede hacer el bosquejo de un sistema en términos de lo que los usuarios intenten hacer con él.

Una de las finalidades del proceso de análisis de un sistema es generar una colección de casos de uso. La idea es tener la posibilidad de catalogar y hacer referencia a esta colección, que sirve como el punto de vista de los usuarios acerca del sistema. Cuando llegue el momento de actualizar el sistema, el catálogo de casos de uso funcionará como un fundamento para obtener los requerimientos de la actualización.

Para los desarrolladores del sistema, ésta es una herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista usuario. Esto es importante si la finalidad de crear un sistema que pueda ser utilizado por la gente en general (no sólo por expertos en computación).

La representación gráfica de los casos de uso es la siguiente: una elipse representa un caso de uso, una figura agregada representa a un actor. El actor que inicia se encuentra a la izquierda del caso de uso, y el que recibe a la derecha. El nombre del actor aparece justo debajo de él, y el nombre del caso de uso aparece ya sea dentro de la elipse o justo debajo de ella. Una línea asociativa conecta a un actor con el caso de uso, y representa la comunicación entre le actor y el caso de uso. Los actores, casos de uso y líneas de interconexión componen un modelo de casos de uso.

La figura siguiente muestra estos símbolos.

Existen dos forma en que los casos de uso se pueden relacionar entre sí. Una de ellas, la *inclusión* le permite volver a utilizar los pasos de un caso de uso dentro de otro. La otra, *extensión* le permite crear un caso de uso mediante la adición de pasos a un existente.

- ***Relaciones entre casos de uso***

Inclusión

Para representar a la inclusión utilizará el símbolo que usó para la dependencia entre clases: una línea discontinua con una punta de flecha que conecta las clases apuntando hacia la clase dependiente. Justo sobre la línea, agregará un estereotipo: la palabra "incluir" bordeada por dos pares de paréntesis angulares.

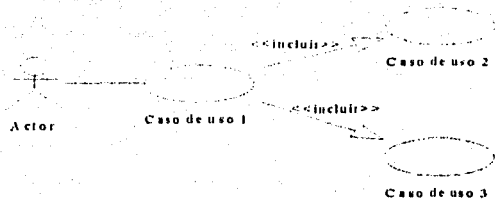
En la figura1 se muestra la relación de <<inclusión>> en el modelo de caso de uso.

Extensión

Para representar la extensión con una línea de dependencia (línea discontinua con punta de flecha), junto con un estereotipo que muestra <<extender>> entre paréntesis angulares. Dentro del caso de uso, el punto de extensión aparecerá debajo del nombre del caso de uso.

En la figura2 se muestra la relación de <<extender>> en el modelo de caso de uso.

Figura1 Relación de <<inclusión>> en el modelo de caso de uso.



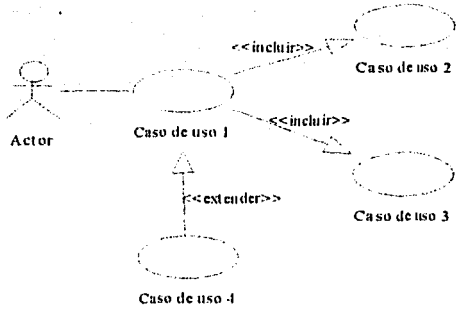


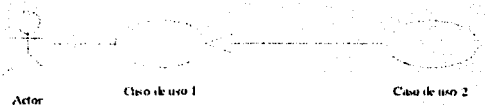
Figura2 Relación de <<extender>> en el modelo de caso de uso.

Generalización

Las clases pueden heredarse entre si y esto también se aplica a los casos de uso. En la herencia de los casos de uso, el caso de uso secundario hereda las acciones y significado del primario, y además agrega sus propias acciones. Puede aplicar el caso de uso secundario en cualquier lugar donde aplique el primario.

Para representar la generalización de casos de uso de la misma forma que lo hace con las clases: con líneas continuas y una punta de flecha en forma de triángulo sin rellenar que

apunta hacia el caso de uso primario, como se muestra en la siguiente figura.



d) Diagrama de estados

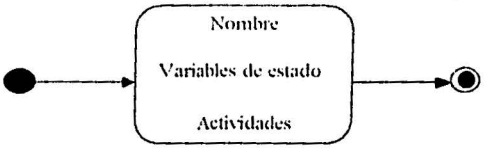
Una manera para caracterizar un cambio en un sistema es decir que los objetos que los componen modifiquen su estado como respuesta a los sucesos y al tiempo.El diagrama de estados UML captura este tipo de cambios. Presenta los estados en los que puede encontrarse un objeto junto con las transacciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambios de estado. Los diagramas que se han presentado modelan el comportamiento de un sistema, o al menos un grupo de clases, objetos o casos de uso. Un diagrama de estados muestra las condiciones de un solo objeto.

La siguiente figura muestra un rectángulo de vértices redondeados que representa a un estado, junto con una línea continua y una punta de flecha, mismas que representan a una transacción. La punta de la flecha apunta hacia el estado donde se hará la transición.

La figura también muestra un círculo relleno que simboliza un punto inicial y la diana que representa un punto final.

El UML le da la opción de agregar detalles a la simbología. Como es poder dividir un símbolo de la clase en tres áreas (nombre, atributos y operaciones). El área superior contendrá el nombre del estado(que tiene que establecer ya sea que haya la subdivisión o no), el área central contendrá las variables de estado, y el área inferior las actividades.

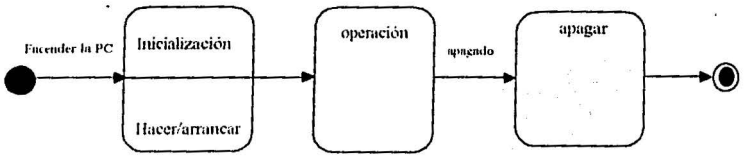
La siguiente figura muestra estos detalles.



Sucesos y acciones

Puede indicar un suceso que provoque una transición (desencadenar un suceso), y la actividad de cómputo (la acción) que se ejecute y haga que suceda la modificación del estado. A los sucesos y acciones los escribirá cerca de la línea de transición mediante una diagonal para separar un suceso desencadenado de una acción. En ocasiones un evento causará una transición sin una acción asociada, y algunas veces una transición sucederá dado que un estado finalizará, se le conoce como transición no desencadenada.

La siguiente figura muestra el diagrama de estados que captura los estados y transacciones en la GUI, cuando encienda su equipo, se ejecutará un proceso de arranque. Al encender la PC se desencadena un suceso que provoca que la GUI aparezca luego de una transición desde el estado de inicialización, y el arranque es una acción que se realiza durante la transición. Como resultado de las actividades en el estado de inicialización, la GUI entra al modo de operación. Cuando desea apagar su PC desencadena un suceso que provoca la transición de estados que captura los estados y transiciones en la GUI.



Importancia

El diagrama de estados del UML proporciona una gran variedad de símbolos y abarca varias ideas (todas para modelar los cambios por los que pasa un objeto). Este tipo de diagrama tiene el potencial de convertirse en algo complejo con asombrosa rapidez.

Es necesario contar con diagramas de estados dado que permiten a los analistas, diseñadores y desarrolladores comprender el comportamiento de los objetos de un sistema. Un diagrama de clases y el diagrama de objetos correspondiente sólo muestra los aspectos estáticos de un sistema. Muestran las jerarquías y asociaciones, y le indican qué son las operaciones. Pero no le muestran los detalles dinámicos de las operaciones.

Estos diagramas aseguran que no tendrán que adivinar lo que se supone harán los objetos. Con una clara representación del comportamiento del objeto, aumenta la probabilidad de que el equipo de desarrollo produzca un sistema que cumpla con los requerimientos.

e) Diagrama de Secuencias

UML le permite expandir su campo de visión y le muestra la forma en que un objeto interacciona con otros. En este campo de visión expandido, incluirá una importante dimensión: el tiempo. La idea primordial es que las interacciones entre los objetos se realizan en una secuencia establecida y que la secuencia se toma su tiempo en ir del principio al fin. Al momento de crear un sistema tendrá que especificar la secuencia y para ello, utilizará el diagrama correspondiente. El diagrama de secuencias consta de objetos que se representan del modo usual: rectángulos con nombre (subrayado), mensajes representados por líneas continuas con una punta de flecha y el tiempo representado como una progresión vertical.

Objeto

Los objetos se colocan cerca de la parte superior del diagrama de izquierda a derecha y se acomodan de manera que simplifiquen al diagrama. La extensión que está debajo (y en

forma descendente) de cada objeto será una línea discontinua conocida como la *línea de vida* de un objeto. Junto con la línea de vida de un objeto se encuentra un pequeño rectángulo conocido como *activación*, el cual representa la ejecución de una operación que realiza el objeto.

La longitud del rectángulo se interpreta como la duración de la activación. La siguiente figura muestra un objeto, su línea de vida y su activación.

Mensaje

Un mensaje que va de un objeto a otro pasa de la línea de vida de un objeto a la de otro. El objeto puede enviarse un mensaje a sí mismo (es decir, desde su línea de vida hacia su propia línea de vida).

Figura que representa la simbología de un mensaje

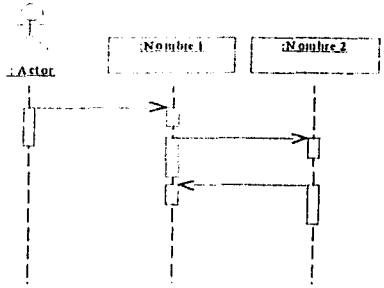
Tiempo

El diagrama representa al tiempo en dirección vertical. El tiempo se inicia en la parte superior y avanza hacia la parte inferior. Un mensaje que esté más cerca de la parte superior ocurrirá antes que uno que éste cerca de la parte inferior.

El diagrama de secuencias tiene dos dimensiones. La dimensión horizontal es la disposición de los objetos, y la dimensión vertical muestra el paso del tiempo.

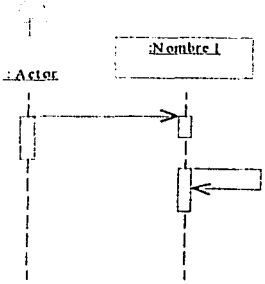
La figura siguiente muestra al conjunto básico de símbolos del diagrama de secuencias, esta

figura muestra a un actor el que normalmente inicia la secuencia, su símbolo no es parte del conjunto de símbolos del diagrama de secuencias.



Representación de la recursividad

En ocasiones un objeto cuenta con una operación que se invoca a sí mismo. A esto se le conoce como *recursividad.* Para representar esto en UML, dibujará una flecha de mensaje fuera de la activación que signifique la operación, y un pequeño rectángulo sobrepuesto en la activación. La siguiente figura representa la recursividad: dibujar una flecha de modo que apunte al pequeño rectángulo, y una que regresa al objeto que inició la recursividad.

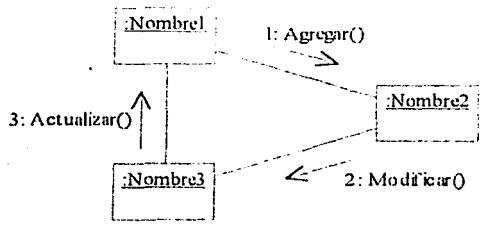


f) Diagrama de Colaboraciones

Los diagramas de colaboraciones muestran la forma en que los objetos colaboran entre sí, tal como sucede en un diagrama de secuencias. De hecho, son semánticamente equivalentes, esto significa que representan la misma información, y podrá convertir un diagrama de secuencias en un diagrama de colaboraciones equivalente y viceversa.

Los diagramas de secuencias destacan la sucesión de las interacciones. Los diagramas de colaboraciones destacan el contexto y organización general de los objetos que interactúan. Otra forma de encontrar diferencia: el diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio.

El diagrama de colaboraciones muestra los mensajes que se envían los objetos entre sí. Para representar un mensaje, dibujará una flecha cerca de la línea de asociación entre dos objetos, esta flecha apunta al objeto receptor. El tipo de mensaje se mostrará en una etiqueta cerca de la flecha; por lo general, el mensaje le indicará al objeto receptor que ejecute una de sus operaciones. El mensaje finalizará con un par de paréntesis, dentro de los cuales colocará los parámetros(en caso de haber alguno) con los que funcionará la operación. Se agregara una cifra a la etiqueta de un mensaje misma que corresponderá a la secuencia propia del mensaje. La cifra y el mensaje se separan mediante dos puntos (:). La figura siguiente muestra la simbología del diagrama de colaboraciones.



g) Diagrama de Actividades

Un diagrama de actividades del UML, es muy parecido a los viejos diagramas de flujo. Le muestra los pasos (conocidos como actividades) así como puntos de decisión y bifurcaciones. Es útil para mostrar lo que ocurre en un proceso de negocios u operación.

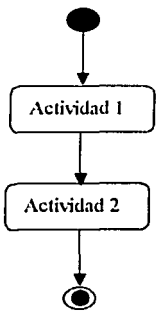
Un diagrama de actividades ha sido diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o proceso. Es una extensión de un diagrama de estados, el diagrama de estados muestra los estados de un objeto y representa las actividades como flechas que conectan a los estados.

El diagrama de actividades resalta, a las actividades.

A cada actividad se le representa por un rectángulo con las esquinas redondeadas (más angosto y ovalado que la representación del estado). El procesamiento dentro de una actividad se lleva a cabo y, al realizarse, se continúa con la siguiente actividad. Una flecha representa la transición de una a otra actividad.

Al igual que el diagrama de estados, el de actividad cuenta con un punto inicial (representado por un círculo relleno) y un final (representado por una diana).

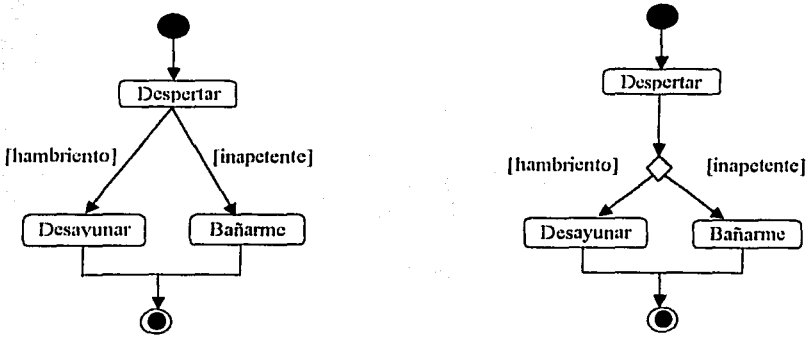
La figura siguiente muestra el punto inicial y final, así como dos actividades y una transición .



Decisiones

Para representar un punto de decisión de una de dos formas: la primera es mostrar las rutas posibles que parten directamente de una actividad y la segunda es llevar la transición hacia un rombo y que de allí salgan las rutas de decisión. De cualquier forma indicará la condición con una instrucción entre corchetes junto a la ruta correspondiente.

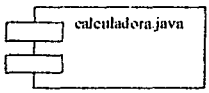
La figura muestra las posibilidades.



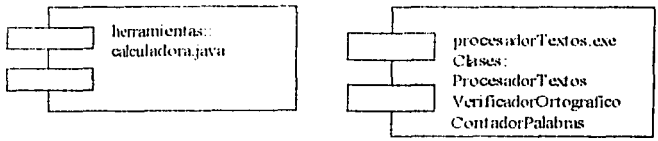
h) Diagrama de Componentes

Un diagrama de componentes contiene, obviamente, componentes, interfaces y relaciones. También puede aparecer otros tipos de símbolos que ya haya visto.

El símbolo principal de un diagrama de componentes es un rectángulo que tiene otros dos sobrepuestos en su lado izquierdo. La siguiente figura muestra éste símbolo. Debe colocar el nombre del componente dentro del símbolo.



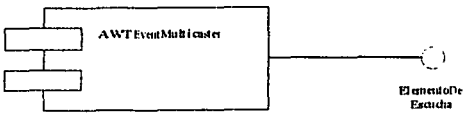
La siguiente figura muestra que si el componente es miembro de un paquete, puede utilizar el nombre del paquete como prefijo para el nombre del componente. También puede agregar información que muestre algún detalle del componente.



Como representar interfaces

Existen dos formas de representar a un componente y sus interfaces: la primera muestra la interfaz como un rectángulo que contiene la información que se le relaciona, se conecta al componente por la línea discontinua y una punta de flecha representada por un triángulo sin rellenar que visualiza la realización.

La siguiente figura muestra la segunda forma de representar a un componente y sus interfaces: esta forma es representativa, ya que representará la interfaz como un pequeño círculo que se conecta al componente por una línea continua.

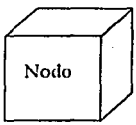


i) Diagrama de Distribución

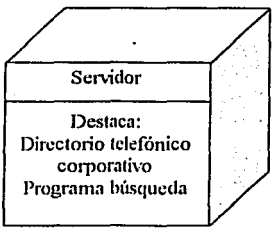
El diagrama de distribución UML muestra la arquitectura física de un sistema informático. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Cada computadora está representada por un cubo y las interacciones entre las computadoras están representadas por líneas que conectan a los cubos.

El diagrama de distribución del UML ilustra la forma en que luce un sistema físicamente cuando sea conjugado. Es posible usar dos tipos de nodos: un procesador, el cual puede ejecutar un componente, y un dispositivo que no lo ejecuta. Normalmente, un dispositivo (como impresora o monitor) tiene contacto de alguna manera con el mundo exterior.

El sistema consta de nodos, donde en UML cada nodo se representa por un cubo. Deberá asignar un nombre para el nodo, y podrá utilizar un estereotipo para indicar el tipo de recurso que sea. La siguiente figura representa un nodo en el UML.

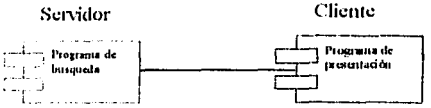


El nombre es una cadena de texto, si el nodo es parte de un paquete, su nombre puede contener también el del paquete. Puede dividir al cubo en comportamientos que agreguen información (como componentes colocados en el nodo), como la figura siguiente:



Una línea asocia a dos cubos y simboliza una conexión entre ellos. Podrá utilizar un estereotipo para dar información respecto a la conexión.

La siguiente figura proporciona ejemplos de conexiones entre nodos.



Los diagramas de UML le permiten examinar un sistema desde distintos puntos de vista. Es importante recalcar que en un modelo UML, no es necesario que aparezcan todos los diagramas. De hecho, la mayoría de los modelos UML contienen un subconjunto de los diagramas que he indicado.

Por lo general, un sistema cuenta con diversas personas implicadas las cuales tienen enfoques particulares en diversos aspectos del sistema. El diseño de un sistema involucra todas las posibles perspectivas, y el programa UML le da una forma de incorporar una perspectiva en particular. El objetivo es satisfacer a cada persona implicada.

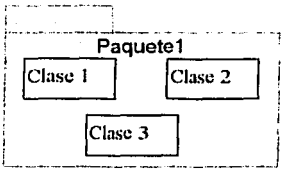
A continuación se describen los elementos que le permiten organizar y extender los diagramas.

1.2.2.2 OTRAS CARACTERISTICAS

a) Paquetes

En algunas ocasiones se encontrará con la necesidad de organizar los elementos de un diagrama en un grupo. Tal vez quiera mostrar que ciertas clases o componentes son parte de un subsistema en particular. Para ello, los agrupará en un paquete, que se representará por una carpeta tabular.

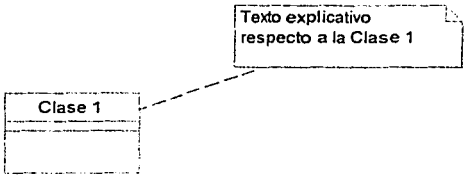
La siguiente figura representa un paquete UML le permite agrupar los elementos de un diagrama.



b) Notas

Es frecuente que alguna parte del diagrama no presente una clara explicación del porqué esta allí o la manera en que trabaja. Cuando éste sea el caso, la nota UML, será útil. La nota es un rectángulo con una esquina doblada, y dentro del rectángulo se coloca la explicación. La nota se adjunta al elemento del diagrama conectándolos mediante una línea discontinua.

La siguiente figura representa que en cualquier diagrama, podrá agregar comentarios aclaratorios mediante una nota.



c) Estereotipos

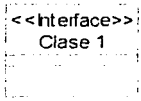
El UML otorga varios elementos de utilidad, pero no es un conjunto minucioso de ellos. De vez en cuando diseñara un sistema que requiera algunos elementos hechos a la medida. Los estereotipos o eclisés le permiten tomar elementos propios del UML y convertirlos en otros.

Lo representa como un nombre entre dos pares de paréntesis angulares y se sitúa encima o

delante del nombre del elemento. Se aplican además de clases a relaciones, nodos, componentes y operaciones.

Una interfaz es una clase que realiza operaciones y que no tiene atributos, es un conjunto de acciones que tal vez quiera utilizar una y otra vez en su modelo. En lugar de inventar un nuevo elemento para representar una interfaz, podrá utilizar un simbolo de una clase con <<Interfaz>> situada justo sobre el nombre de la clase.

Ejemplo de notación de estereotipo

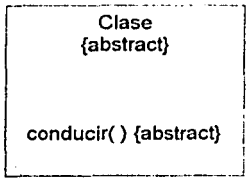


d) Clase abstracta

Una clase abstracta es una clase no puede crear instancias (objetos). Una clase abstracta sólo se utiliza para heredar de ella atributos y comportamientos comunes. Una clase se puede especificar explícitamente como abstracta poniendo la etiqueta {abstracta} dentro de la banda de nombre de la clase y debajo del mismo.

Una clase abstracta se representa con el nombre en cursiva (itálica). Cualquier operación se indica también en cursiva {itálica} y a su derecha se indica también la expresión {abstract}.

Ejemplo de notación de clase abstracta



1.2.2.3 RELACIONES

Las relaciones que se pueden utilizar en UML son asociaciones, generalizaciones y dependencias[3].

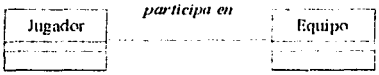
a) Asociaciones

Una asociación representa una relación estructural entre objetos de clases diferentes.

Una asociación representa que objetos de dos clases tienen un enlace entre ellos, significando, por ejemplo, que <<unas clases conocen de las otras clases>>, <<están conectadas a>>, <<por cada objeto x hay un objeto y>>, etc.

En esencia, una asociación es una relación entre clases que especifica cómo se pueden enlazar juntas las instancias de las clases. La conexión entre una asociación y los enlaces correspondientes se realiza etiquetando ambas asociaciones y enlazándolas con el mismo nombre. Un enlace es la relación entre objetos. Una asociación se representa por una línea que une dos clases; el nombre de la asociación se escribe en la línea.

Representación de una asociación



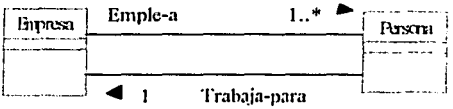
La asociación tiene un nombre (cerca de la línea que representa la asociación), con frecuencia un verbo, aunque también se permiten nombres. Para expresar cuántas instancias existen se utiliza la multiplicidad o cardinalidad.

La multiplicidad representa el número de objetos (rango) que indica cuántos objetos se pueden enlazar. El rango pueden ser cero-a-uno (0..1), cero a muchos (0..* o sólo *) el

[3] JOYANES AGUILAR LUIS.: Programación Orientada a Objetos segunda edición, 1998, págs: 113-148.

símbolo * indica muchos, unos-a-muchos (1..*), cuatro(4), seis o doce elementos (6,12), o incluso varios números (2,5,7 8,9-15). Si no se especifica multiplicidad, se considera 1 por omisión. La multiplicidad se muestra en los extremos de la asociación, en las clases donde es aplicable.

Ejemplo: Trabaja-para es el nombre de la asociación Persona a Empresa. Emplea-a es el nombre de la asociación Empresa a Persona. El triángulo indica la dirección.

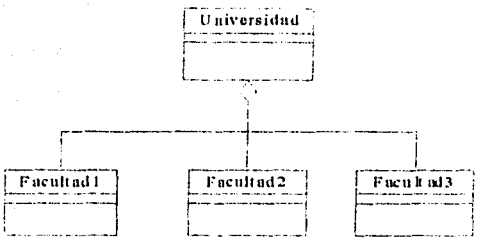


b) Agregación

La agregación es un caso especial de asociación. El agregado indica que la agregación entre clases es un tipo de todo-parte. Las palabras o frases que se utilizan para identificar agregados son es-parte-de, consta-de, tiene-un, contiene, etc. Existen diferentes tipos de agregación: agregación compartido y agregación composición.

El agregado se muestra con un pequeño rombo conectado a un extremo de una línea (en el lado todo, la clase que contiene a la otra clase, el lado parte). La agregación es un caso especial de asociación con indicación de multiplicidad, roles y cualificadores. Un agregado puede implementarse de igual modo que una asociación, esto es, algún tipo de referencia de objetos del todo a las partes, y viceversa.

Ejemplo: Una universidad tiene varias facultades o escuelas, de modo que se puede suprimir o añadir alguna facultad y seguirá existiendo la universidad como tal.



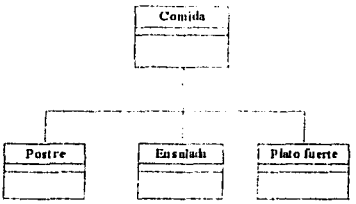
Agregación Compartida

Una agregación compartida es una en las que las partes pueden ser partes en cualquier todo.

Por ejemplo, un equipo se compone de diferentes miembros(personas). Una persona puede ser miembro de muchos equipos, es decir, las personas son las partes compartidas.

La agregación es compartida si la multiplicidad en el todo es distinta de uno.

Esta agregación se muestra con una línea y un rombo sin rellenar conectado a la parte todo.



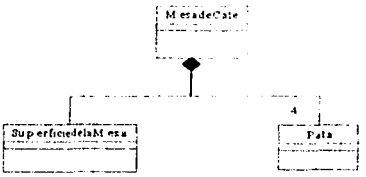
TESIS CON
FALLA DE ORIGEN

Agregación de Composición

Una agregación de composición es aquella que posee(contiene) a sus partes y entraña una fuerte dependencia de propiedad. Las partes viven en el todo, de modo que se destruyen cuando se destruye el todo. La multiplicidad en el lado todo debe ser cero o uno, pero la multiplicidad en el lado parte puede ser un intervalo.

Esta agregación se muestra con una línea y un rombo sólido (dibujado en negro) conectado a la parte todo.

Ejemplo:



c) Generalización

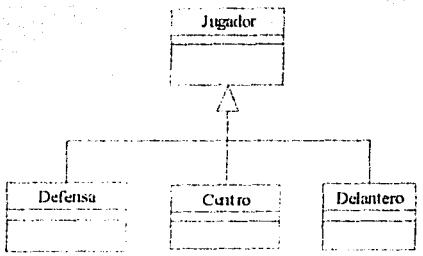
Generalización en UML es una relación entre un elemento más general y uno más específico. La clase específica, denominada subclase, hereda de la clase general, denominada superclase. Se heredan los atributos, las operaciones y todas las relaciones. Los atributos y operaciones con la visibilidad pública en la superclase serán pública en la subclase. Los miembros (atributos y operaciones) que tienen visibilidad privada se heredarán, pero no son accesibles dentro de la subclase.

La generalización se representa por una punta de flecha y una línea se dirige de la clase especialización a la clase generalización. Una generalización se conoce como relación <<es-un>>, <<es-un-tipo-de>> y es una relación que es capaz de decir es-un entre el elemento especializado y el elemento general.

Por ejemplo, la clase Cuenta modela las características comunes compartidas por los tres

tipos de cuenta especializada.

Ejemplo



1.2.3 BENEFICIOS DE LA TECNOLOGÍA A OBJETOS

Una vez que se ha implantado una tecnología en la organización (y que se ha superado la curva de aprendizaje) tenemos:

1. Mejores tiempos totales de desarrollo. En la mayoría de organizaciones hoy en día, el tiempo que pasa desde que un proyecto arranca hasta que se estabiliza es más del doble de lo planeado originalmente. Con el uso de UML las fases de análisis y diseño consumirán mayor tiempo, pero el tiempo de construcción, implantación y estabilización se reducen drásticamente debido a que no hay correcciones mayores en las fases de mayor impacto de un proyecto.
2. Mayor calidad. El uso de UML hace indispensable la participación del usuario en la definición de requerimientos y por lo tanto mejora considerablemente el apego del sistema a las necesidades de sus usuarios. El mantenimiento correctivo se reduce drásticamente (hasta un 80 % con respecto a un sistema hecho sin metodología). Algo similar ocurren

en los proyectos de reingeniería.

3. Mejor soporte a la planeación y al control de proyectos. A existir entregables definidos y estadarizados en las distintas fases de un proyecto al ser éstos revisables y certificables en las distintas fases de un proyecto y al ser éstos trabajo pueden ser fácilmente creados y corroborados en avance. Lo que permite tomar decisiones a tiempo.
4. Mayor independencia del personal de desarrollo. Al tener documentadas las aplicaciones en un lenguaje estándar, podemos mover al personal de una aplicación a otra sin correr altos riesgos y sin depender del conocimiento personal de las aplicaciones.
5. Mayor soporte al cambio organizacional, comercial y tecnológico. Un modelo permite cuantificar el impacto de un cambio antes de hacerlo y permite ensayar distintos enfoques de solución. Con UML un cambio se puede hacer primero en papel.
6. Alto reuso. Los productos de un desarrollo pueden ser usados en otro. Se pueden crear componentes reusables que con la difusión y administración adecuadas minimizarán costos y errores.
7. Minimización de costos. Los punto antes mencionados tienen un impacto económico que generalmente tiende a ser proporcional al tamaño de la organización.

1.2.4 SERVICIOS NECESARIOS PARA IMPLANTAR ESTA TECNOLOGÍA.

Aunque varía un poco de organización a organización los servicios de apoyo necesarios para la implantación de esta tecnología, podemos mencionar los siguientes:

–Consultoría para la Planeación. Cuando las áreas involucradas son muchas, el impacto de la introducción de esta tecnología requerirá ser adecuada, entre proceso deber ser hecho por la organización y apoyado por un equipo con experiencia en la administración de este cambio.

–Capacitación. Las técnicas involucradas pueden ser aprendidas directamente de los libros y manuales de UML, sin embargo el tiempo necesario puede ser prohibitivo. Un servicio de capacitación de alta calidad generará la cultura básica para el óptimo aprovechamiento de la tecnología. Capacitación en UML, Análisis y Diseño de aplicaciones es sugerida.

–Mentoring. El desarrollo de proyectos pilotos, documentación o reingeniería deben ser apoyados por uno o más expertos en el uso de UML que aseguren que el equipo adquiere el conocimiento práctico del uso de UML y agilicen su uso.

–Control de calidad. Una vez que un equipo ya ha aprendido el uso de UML es sano contar con un staff de control de calidad externo (y experto) que certifique la calidad de los productos y genere gente con éste perfil hacia el interior de la organización. Este servicio también puede ser útil para controlar la calidad de los desarrollos efectuados por empresas externas.

–CASE. Una herramienta automatizada facilitará el uso de UML y proporcionará un mecanismo de control de documentación. Con interfaces hacia distintas herramientas de desarrollo, reducirá el tiempo de implementación y con módulos de ingeniería reversa, facilitará procesos de mantenimiento, reingeniería afinación de aplicaciones.

Existen varias aplicaciones sobre distintas plataformas, que generan la documentación en base a la metodología UML. Las cuales poseen ciertas características además tener un costo para adquirirlas.

En nuestro caso hicimos utilizamos una herramienta llamada Rational Rose 98 Professional, que corre bajo Windows 95 ó superior, la cual, es una versión beta. Con está aplicación se generaron los diagramas necesarios para el desarrollo del sistema.

Estos diagramas se podrán consultar en una liga que se encuentra dentro del sistema generado, como referencia de la documentación del sistema.

CAPITULO 2. CONCEPTOS DE BASE DE DATOS

2.1 INTRODUCCION

Las bases de datos constituyen una parte integral y fundamental de un sistema de información y tienen su razón de ser en la misma existencia de éste, por lo que, nos ha parecido oportuno introducir el estudio de las bases de datos.

El problema de la información está estrechamente relacionado con el desarrollo económico y social. La investigación, la planificación y la toma de decisiones exigen una información precisa, oportuna, completa, coherente y adaptada a las necesidades específicas de cada usuario y de cada circunstancia. Aunque a veces el deseo de información es un fin en sí mismo, en general responde a una necesidad de conocer el entorno socioeconómico y cultural en el que nos desenvolvemos con fines de investigación y de toma de decisiones. La disponibilidad de información es precisa también para que el individuo pueda participar en los asuntos públicos.

Son muchos los factores que han influido en la transformación que se han operado en el papel que desempeña la información en los contextos económicos y social. Entre ellos es preciso destacar la elevación del nivel cultural, el afán de desmasificación que lleva a una mayor diversidad, con el consiguiente crecimiento de las necesidades de información, el deseo de participar en las decisiones públicas, las exigencias de planificación y ordenación del territorio, las tendencias hacia una descentralización, que requiere datos cada vez más detallados para áreas más pequeñas, la aparición de nuevos métodos para la toma de decisiones, etc.

A medida que se toma conciencia del valor de la información, se manifiesta la necesidad, que

llega a ser imperiosa, de impulsar la libre circulación de la información, tanto en un plano nacional como internacional, respondiendo así al mandato de las normas que establecen la libertad de información como uno de los grandes derechos de la civilización actual.

2.1.1 Cualidades de la Información

La explosión de la información, como se llama a veces a este enorme crecimiento de las necesidades de la información y a la mayor disponibilidad de este recurso, a una polución informativa. Fenómeno análogo al de la contaminación del aire, en el que la información, al perder sus cualidades, no puede cumplir sus objetivos, llegando incluso a ser más nocivo que beneficiosa para sus destinatarios. Para evitar lo peligroso de la polución informativa se debe exigir a la información un conjunto de cualidades que mantengan su valor comunicativo, ya que para hacer honor a su nombre debe ser capaz de informar, es decir, de aportar un conocimiento.

Las cualidades que debe poseer la información, y que hacen de ella un recurso fundamental de las organizaciones y de los individuos, son básicamente: precisión, oportunidad, compleción, significado e integridad, todas ella en el grado que exija cada sistema concreto.

La precisión[1], se puede definir como el porcentaje de información correcta sobre la información total del sistema (fichero, base de datos, etc.). De todas formas, el usuario ha de tener presente que el manejo por computadora no puede mejorar la calidad de los datos que son elaborados, lo único que puede hacer la computadora es señalar ciertos errores o incompatibilidades, e incluso sustituir el dato detectado como erróneo aparente, es decir, que

[1] Algunos autores distinguen entre "exactitud" y "precisión", refiriéndose con "exactitud" a la ausencia de errores de transmisión o de cálculo, y con "precisión" al grado de aproximación entre la información almacenada o accedida y el valor real. En estadística se utiliza el término acuracidad para indicar el conjunto de precisión, dispersión escasa, e insesgamiento, proximidad al valor real.

sea coherente. En resumen, si queremos que los resultados de la computadora sean precisos, debemos también suministrarle datos precisos, no pudiendo pretender en los resultados una precisión superior a la que tenían los datos de entrada. Una precisión baja lleva a una falta de credibilidad del usuario hacia la información que se le proporciona.

La oportunidad, se refiere al tiempo transcurrido desde el momento en que se produjo el hecho que originó el dato hasta el momento en el que la información se pone a disposición del usuario. Otras veces la oportunidad se mide en función del tiempo transcurrido desde que el dato tendría que estar disponible, o bien respecto al desfase que produce el proceso por ordenador. Al igual que ocurre con la precisión, también la oportunidad depende de cada aplicación. Por ejemplo, para un censo en el cual se manejan millones de datos de carácter bastante estable, un tiempo de proceso de meses no le resta oportunidad a la información, en cambio esta demora en la obtención de los indicadores de coyuntura, sería inadmisible.

En general, el valor de la información va disminuyendo con el transcurso del tiempo[2], e incluso, después de cierto momento, puede llegar a perder totalmente la relevancia que pudiera tener, la pérdida de valor será más o menos dependiendo del tipo de información.

Otra cualidad que ha de tener la información es la compleción, lo que significa que ha de ser completa para poder cumplir sus fines. Por ejemplo, un informe que se emite con el objeto de que un directivo tome una decisión, ha de contener todos los elementos informativos necesarios para apoyar dicha decisión. La compleción absoluta es imposible de conseguir, y lo que se suele pretender en los sistemas de información es alcanzar un nivel que se considere suficiente, el cual dependerá de dos factores: de los datos existentes en el sistema de información y de los que el sistema es capaz de localizar ante una consulta concreta. En

[2] En investigaciones históricas, por el contrario, la información gana con el transcurso del tiempo.

este segundo factor influirá la flexibilidad e idoneidad del lenguaje de recuperación y el acierto en la formulación de la consulta. Así pues, la compleción no es sólo función de la información en sí misma, sino también de otros factores, tanto técnicos como humanos.

La información que se suministra al usuario debe ser también significativa; es decir, ha de poseer el máximo contenido semántico posible, ya que sin el no constituiría verdadera información. Esto lleva a que ha de ser comprensible e interesante, lo que supone no proporcionar a los usuarios grandes masas de información que por su volumen no puedan ser asimiladas. Un volumen de información justo es condición indispensable para que ésta sea significativa. Cuando se realiza el diseño de un sistema es preciso tener en cuenta que la información suministrada por este ha de ser, además de fácilmente interpretable, sólo la necesaria y suficiente para que se cumplan los fines propuestos.

Asimismo toda la información contenida en el sistema debe ser coherente en si misma, además de consistente con las reglas semánticas propias del mundo real al que ha de representar lo más fielmente posible, esta cualidad, que en las bases de datos se conoce a veces con el nombre de integridad, coincide en parte con el concepto que hemos definido como precisión.

Es preciso también atender a la seguridad de la información, ya que ésta ha de ser protegida tanto frente a su deterioro por causa físicas o lógicas como frente a accesos no autorizados. La seguridad de la información esta adquiriendo una gran relevancia, muy especialmente con la difusión de las nuevas posibilidades de las comunicaciones y la enorme extensión de redes de conexión como internet e intranet. Actualmente el concepto de seguridad comprende confidencialidad, diponibilidad e integridad.

Cuando se están haciendo los estudios que nos llevarán a la implantación de un sistema de información, es preciso tener muy en cuenta todos estos requisitos de la información

buscando el punto de equilibrio que permita alcanzar los objetivos del sistema a un coste aceptable, ya que cuentas más cualidades reúnan la información más se incrementará su coste de obtención y tratamiento.

Por otro lado, unas cualidades pueden resultar incompatibles con otras, así pretender una gran precisión lleva consigo generalmente una pérdida de oportunidad. Por lo que, es necesario llegar a una solución de compromiso, encontrando el punto de equilibrio entre las diversas cualidades de la información, dentro de unos objetivos concretos de cada sistema y siempre a un costo aceptable.

2.1.2 Concepto de Base de Datos

Son muy numerosas las definiciones de base de datos, y si se analizan determinadamente, se suele observar en casi todas ellas coincidencias en ciertos elementos, aunque también se detecta la falta de otros fundamentales, o al menos muy importantes, que son característicos de las bases de datos y que marcan la diferencia entre este concepto y el de ficheros.

La aparición de la expresión base de datos se produce a comienzos de los años sesenta. En 1963 tuvo lugar en Santa Mónica (E. U.) un simposio en cuyo título se encontraba la expresión Data Base. En una de sus sesiones, se propuso una definición de base de datos que, según las actas del simposio, no fue universalmente aceptada. Posteriormente, en 1967, el grupo de estandarización Codasyl decidió cambiar su primitiva denominación en la que no aparecía base de datos por el de Data Base Task Group. Poco a poco, el concepto y la expresión base de datos iba imponiéndose.

Nosotros vamos a empezar estudiando las características esenciales de una base de datos, para apoyar en ellas el concepto de base de datos.

En primer lugar, y en esto coinciden todas las definiciones, una base de datos es un

conjunto, colección o depósito de datos almacenados en un soporte informático no volátil. Los datos están interrelacionados y estructurados de acuerdo con un modelo capaz de recoger el máximo contenido semántico. Dada la relevancia que tienen en el mundo real las interrelaciones entre los datos, es imprescindible que la base de datos sea capaz de almacenar estas interrelaciones. En el mundo real existen, además, restricciones semánticas, a las que se está concediendo una importancia creciente y que, en los sistemas actuales, tienden a almacenarse junto con los datos, al igual que ocurre con las interrelaciones. La base de datos se describe y se manipula apoyándose en un modelo de datos.

La redundancia de los datos debe ser controlada, de forma que no existan duplicidades perjudiciales ni innecesarias, y que las redundancias físicas, convenientes muchas veces a fin de responder a objetivos de eficiencia, sean tratadas por el mismo sistema, de modo que no puedan producirse inconsistencias. Esto podría resumirse diciendo que en las bases de datos no debe existir redundancia lógica, aunque si se admite cierta redundancia física por motivos de eficiencia. Por tanto, un dato se actualizará lógicamente por el usuario de forma única, y el sistema se preocupará de cambiar físicamente todos aquellos campos en los que el dato estuviese repetido en caso de existir redundancia física, es lo que se denomina también redundancia controlada por el sistema.

Las bases de datos pretenden servir al conjunto de la organización, manejando los datos como otro recurso que viene a añadirse a los ya tradicionales. Por tanto, las bases de datos han de atender a múltiples usuarios y a diferentes aplicaciones, en contraposición a los sistemas de ficheros, en los que cada fichero esta diseñado para responder a las necesidades de una determinada aplicación.

Otro aspecto importante de las bases de datos es la independencia, tanto física como lógica,

entre datos y tratamientos. Esta independencia, objetivo fundamental de las bases de datos, es una característica esencial que distingue las bases de datos de los ficheros y que teniendo una enorme influencia en la arquitectura de los SGBD.

La definición o descripción del conjunto de datos contenidos en la base, lo que se denomina estructura o esquema de la base de datos, deben ser únicas y estar integradas con los mismos datos. En los sistemas basados en ficheros, los datos se encuentran almacenados en ficheros, mientras su descripción está separada de los mismos, formando parte de los programas, para lo cual se precisa que los lenguajes faciliten medios para la descripción de los datos. Suele haber además una documentación adicional, habitualmente en soporte de papel, en general insuficiente y desactualizada. Este tipo de organización da origen a infinidad de problemas, ya que a veces no se sabe cuál es la descripción de un determinado fichero, bien por pérdida de la misma, bien porque no se ha actualizado debidamente la correspondiente documentación y tampoco se conoce exactamente el programa que lo trataba. En las bases de datos, la descripción, y en algunos casos también una definición y documentación completas, se almacenan junto con los datos de modo que éstos están autodocumentados, y cualquier cambio que se produzca en dicha documentación se ha de reflejar y quedar recogido en el sistema, con todas las ventajas que de ese hecho se derivan.

La actualización y recuperación de los datos debe realizarse mediante procesos bien determinados, incluidos en el SGBD, el cual ha de proporcionar también instrumentos que faciliten el mantenimiento de la seguridad del conjunto de datos.

El concepto de base de datos ha ido cambiando y configurándose a lo largo del tiempo, en la actualidad y de acuerdo con estas características que acabamos de analizar, podemos definir la base de datos como: Colección o depósito de datos integrados, almacenados en soporte secundario y con redundancia controlada. Los datos, que han de ser compartidos por

diferentes usuarios y aplicaciones, deben mantenerse independientes de ellos y su definición única y almacenada junto con los datos, se ha de apoyar en un modelo de datos, el cual ha de permitir captar las interrelaciones y restricciones existentes en el mundo real. Los procedimientos de actualización y recuperación comunes y bien determinados, facilitarán la seguridad del conjunto de datos[3].

El Sistema de Gestión de Bases de Datos (SGBD), es el conjunto de programas que permiten la implantación, acceso y mantenimiento de la base de datos. El SGBD junto con la base de datos y con los usuarios, constituyen el Sistema de Base de Datos.

2.1.3 Distintos niveles de abstracción de una Base de Datos

En los Sistemas de Información, existen dos estructuras la lógica (vista del usuario) y la física (forma en que se encuentran los datos en el almacenamiento). En las bases de datos aparece un nuevo nivel de abstracción que se ha denominado de diversas maneras: nivel conceptual, lógico global, etc. Esta estructura intermedia pretende una representación global de los datos que se interpongan entre las estructuras lógica y física de la arquitectura a dos niveles, siendo independiente, tanto del equipo como de cada usuario en particular.

La estructura lógica de usuario o esquema externo es la visión que tiene de la base de datos cada usuario en particular, la estructura lógica global (también denominada esquema conceptual) responde al enfoque del conjunto de la empresa y la estructura física (o esquema interno) es la forma en que se organizan los datos en el almacenamiento físico. La estructura de una base de datos en estos tres niveles de abstracción tiene como principal objetivo conseguir la independencia entre datos y aplicaciones.

---

[3] CASTAÑO ADORACION DE MIGUEL, PIATTINI MARIO.: Fundamentos y Modelos de Bases de Datos, México, 1993, Alfaomega, 2ª Edición, pág. 28.

Estructura lógica de usuarios (Esquema externo). Debido a que un esquema externo es la visión que de la base de datos tiene un usuario en particular, en él deberán encontrarse reflejados sólo aquellos datos e interrelaciones que necesite el correspondiente usuario. También habrán de especificarse las restricciones de uso, como puede ser el derecho a insertar o a borrar determinados datos o el acceso a los mismo, etc. Asimismo y aunque esto no sea lo más conveniente, ya que indica una dependencia física-lógica, puede que aparezcan en este nivel los caminos de acceso a los datos, hecho que dependerá en gran medida del modelo de datos en el que se apoya el correspondiente SGBD, en el modelo relacional los caminos de acceso sólo se encuentran en el nivel interno, no siendo nunca visibles por los usuarios.

Habrá tantos esquemas externos como exijan las diferentes aplicaciones. Un mismo esquema externo podrá ser utilizado por varias aplicaciones.

Estructura Lógica Global (Esquema Conceptual). En el esquema conceptual, por ser la visión global de los datos, deberá incluirse la descripción de todos los datos e interrelaciones entre estos, así como las restricciones de integridad y de confidencialidad. La estabilidad de estos conceptos disminuye en el orden en el que los hemos citado. Así, las restricciones de confidencialidad serán menos estables que las de integridad, y éstas, a su vez, serán menos estables que las interrelaciones o que los datos. Por esta razón, algunos autores proponen que este esquema se divida en varios, uno para cada concepto, de modo que por ejemplo, un cambio en las restricciones no lleve consigo una nueva definición de todo el esquema.

Estructura Física (Esquema Interno). Aunque el contenido del esquema interno depende mucho de cada SGBD podemos distinguir tres clases de aspectos que deben especificarse

en el.

Estrategia de almacenamiento. Es este apartado se incluye la asignación de espacios de almacenamiento para el conjunto de datos. También deberá indicarse la estrategia de emplazamiento de los datos que han sido utilizada para optimizar tiempos de respuesta y espacio de memoria secundaria, por último deberán aparecer aspectos como el tratamiento de los desbordamientos, etc.

Caminos de acceso. Incluimos en los caminos de acceso la especificación de claves, así como la de indices o punteros.

Miscelánea. Además de los aspectos citados, habría que incluir, en el esquema interno, otros varios, como técnicas de compresión de datos, de criptografiado, la correspondiente entre esquema interno y esquema conceptual, técnicas de ajuste o afinamiento, optimización, etc.

El administrador de la base de datos habrá de especificar:

Dispositivos de memoria: Tamaño de la página, numero de páginas asignadas a cada área de almacenamiento, tamaño de las áreas de estrada/salida (buffers), etc.

Correspondencia entre esquemas (mapping): Por omisión, se suele suponer que existe una correspondencia uno a uno entre los registros del esquema conceptual y los registros almacenados, en caso contrario, el administrador debe indicar la relación existente entre ellos.

Organizaciones fisicas: Para mejorar la recuperación y los tiempos de acceso, el sistema debe dar facilidades para que el administrador defina el tipo de organización (dispersión, agrupamientos, indices, etc) que considere más adecuada a fin de lograr la máxima eficiencia, dependiendo del SGBD podrá también definir punteros entre registros, privilegiados así determinados caminos de acceso.

2.2 MODELOS DE DATOS

Para describir la estructura de una BD es necesario definir el concepto de modelo de datos, una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia. Los diversos modelos de datos se dividen en tres grupos : modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos de datos[4].

2.2.1 Modelos lógicos basados en objetos

Los modelos lógicos basados en objetos se usan para describir datos en el nivel conceptual y de visión. Se caracterizan porque proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Hay muchos modelos diferentes, y aparecerán más.

Algunos de los más conocidos son el modelo entidad-relación, el orientado a objetos, el binario, el semántico de datos, el infológico y el modelo funcional de datos. El modelo entidad-relación ha ganado aceptación y el cual se utilizó para modelar la Base de Datos del Sistema de Seguimiento a Egresados, el modelo orientado a objetos incluye muchos conceptos del anterior, y esta ganado aceptación rápidamente.

2.2.1.1. El modelo entidad-relación

El modelo de datos entidad-relación (**E**-R) se basa en una percepción de un mundo real que consiste en una colección de objetos básicos llamados entidades, y relaciones entre estos objetos. Una entidad es un objeto distinguible de otros por medio de un conjunto específico

[4] KORTH HENRY F., SILBERSHATZ ABRAM.: Fundamentos de Bases de Datos, México, 1993, McGrawn-Hill, 2ª Edición, pág.6

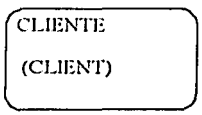
de atributos. Por ejemplo, los atributos número y saldo describen una cuenta particular. Una relación es una asociación entre varias entidades. Por ejemplo, una relación CliCta asocia a un cliente con cada cuenta que posee. El conjunto de todas las entidades del mismo tipo y relaciones del mismo tipo se denomina conjunto de entidades y conjunto de relaciones. Además el modelo E-R representa ciertas restricciones a las que deben ajustarse los contenidos de una BD. Una restricción importante es la cardinalidad de asignación, que expresa el número de entidades a las que puede asociarse otra entidad mediante un conjunto de relación. La estructura lógica global de una BD puede expresarse gráficamente por un diagrama E-R.

a) Entidad. Es una persona, lugar, cosa, concepto o suceso, real o abstracto, de interés para la empresa. Es aquel objeto acerca del cual queremos almacenar información en la base de datos.

Llamaremos tipo de entidad a la estructura genérica y ocurrencia de entidad a cada una de las realizaciones concretas de este tipo de entidad. Así, el tipo de entidad AUTOR se refiere a la estructura que nos describe las características de los autores como una abstracción, mientras que una ocurrencia de autor será cada uno de los autores en concreto.

Estándares de diagramación:

- Cajas de cualquier dimensión con las esquinas redondeadas.
- Un nombre único para cada entidad.
- Nombre de la entidad en mayúsculas.
- Nombre del sinónimo, entre paréntesis (opcional).



Identificar y modelar entidades

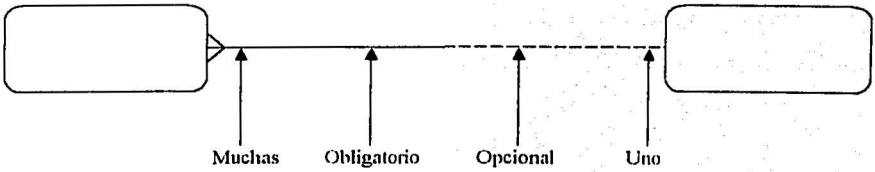
- Examinar los sustantivos. ¿Son aspectos importantes?
- Poner un nombre a cada entidad.
- ¿Existe información de interés para la compañía acerca de la entidad?
- ¿Es cada instancia de la entidad identificable de manera única? ¿Cuál o cuáles atributos sirven como UID?
- Escribir la descripción de la entidad. "Un empleado tiene como significado ser un EMPLEADO pagado por la compañía. Por ejemplo, Javier Díaz y María Herrera son empleados".
- Diagramar cada entidad y sus atributos.

b) Relación. Definimos la relación como la asociación o correspondencia entre entidades. Llamamos Tipo de relación a la estructura genérica del conjunto de interrelaciones existentes entre dos o más tipos de entidad, mientras que la ocurrencia concretas de cada uno de los tipos de entidad que intervienen en la interrelación. Así, el tipo de entidad AUTOR se interrelacionan con el tipo de entidad DOCUMENTO mediante el tipo de interrelación Escribe. Una relación es bidireccional, representa la asociación entre dos entidades, o entre una entidad consigo misma.

Sintaxis de relación:

Cada entidad1 { Debe ser / o / Puede ser } Nombre de la relación { Una o más / o / una y solamente } Entidad2

Estándares de diagramación

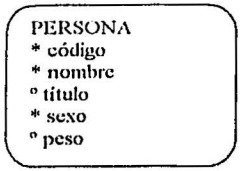


- Determinar si existe una relación.
- Nombrar cada dirección de la relación.
- Determinar la opcionalidad de cada dirección de la relación.
- Determinar el grado de cada dirección de la relación.
- Leer en voz alta las relaciones para validarlas.

c) Atributo. Es cada una de las propiedades o características que tiene un tipo de entidad o de interrelación. Así, el tipo de entidad AUTOR tiene como atributos el Nombre, la Nacionalidad, Fecha_nac, la Biografía, etc.; y los atributos del tipo de entidad DOCUMENTO son, entro otros, Título y Resumen.

Los atributos son información acerca de una entidad que necesitamos conocer o tener.

Estándares de diagramación



- Los nombres de los atributos están en singular, en minúsculas, y no incluyen el nombre de las entidades.
- Marcas de atributos: * para obligatorios y
  º para opcional.

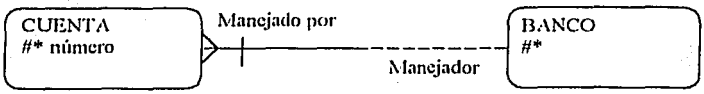
Analizar y modelar atributos

- Identificar los candidatos para atributos.
- Asociar cada atributo con una entidad.
- Poner un nombre a cada atributo.
- Determinar la opcionalidad de los atributos.
- Validar que los atributos sean realmente atributos y no entidades.
- Descomponer los atributos a su menor componente.
- Verificar que cada atributo tenga un solo valor.
- Verificar que los atributos no sean derivados.

Cada entidad debe ser identificable de manera única. Un identificador único (UID) es cualquier combinación de atributos y/o relaciones que sirven para identificar de manera única a cada ocurrencia de una entidad.

Estándares de Diagramación.

- El # indica que un atributo es parte del UID de una entidad.



- La barra UID indica que una relación es parte del UID.

Identificar los UID para cada cantidad.

- Buscar atributos candidatos para ayudar a identificar una entidad.
- Determinar la dependencia de entidades sobre otras entidades relacionadas.
- Definir el UID para la entidad.

2.2.1.1.1 Normalizar el Modelo de Datos

Normalizar es un concepto del Modelo Relacional, pero sus principios se aplican al Modelo Entidad Relación.

Validar cada atributo usando las reglas de normalización.

Regla de Forma Normal	Descripción
Primera Forma Normal (1FN)	Todos los atributos deben tener un solo valor para cada instancia.
Segunda Forma Normal (2FN)	Un atributo debe ser dependiente del identificador único completo
Tercera Forma Normal (3FN)	Ningún atributo no-UID puede ser dependiente de otro atributo no-UID.

2.2.1.2. El modelo orientado a objetos.

Al igual que el modelo E-R, el modelo orientado a objetos se basa en una colección de objetos. Un objeto contiene valores acumulados en variables de instancia dentro de él, y estos valores son objetos por si mismos. Así, los objetos contienen objetos a un nivel de anidamiento arbitrario. Un objeto también contiene partes de código que operan sobre el objeto, que se denominan métodos.

Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan en clases. Una clase puede se vista como una definición de tipo para objetos. La única forma en la que un objeto puede acceder a los datos de otro objeto es invocando a un método de ese otro objeto. Esto se llama envío de un mensaje al objeto. Así, la interfaz de llamada de los métodos de un objeto define su parte visible externamente, la parte interna del objeto (las variables de instancia y el código de método) no son visibles externamente. El resultado es dos niveles de abstracción de datos.

Para ilustrar el concepto, considérese un objeto que representa una cuenta. Dicho objeto contiene las variables instancia número y saldo, y el método interés de pago, que añade interés al saldo. Supóngase que se paga el 6% en todas las cuentas, pero ahora se va a pagar el 5% si el saldo es menor que $1000, y el 6% si es mayor o igual. Bajo la mayoría de los modelos de datos, esto implicaría cambiar de código en uno o más programas de aplicación. Bajo este modelo, sólo se hace un cambio dentro del método interés de pago. El interfaz externo del objeto permanece sin cambios.

El modelo orientado a objetos se basa en encapsular código y datos en una única unidad, llamada objeto. El interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes.

2.2.2. Modelos lógicos basados en registros.

Los modelos lógicos basados en registros se utilizan para describir datos en los modelos conceptual y físico. A diferencia de los modelos lógicos basados en objetos, se usan para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de la implementación.

Los modelos basados en registros se llaman así porque la BD está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija. La estructura más rica de estas BD a menudo lleva a registros de longitud variable en el nivel físico.

Los modelos basados en registros no incluyen un mecanismo para la representación directa de código de la BD, en cambio, hay lenguajes separados que se asocian con el modelo para expresar consultas y actualizaciones Los tres modelos de datos más aceptados son los modelos relacional, de red y jerárquico. El modelo relacional ha ganado aceptación por encima de los otros.

2.2.2.1 Modelo relacional.

Tras la introducción del modelo relacional se ha desarrollado una teoría esencial para las BD relacionales, que ayuda al diseño de BD relacionales y al procesamiento eficiente de solicitudes de información. El modelo relacional se ha establecido como el principal modelo de datos para aplicaciones comerciales de procesamiento de datos.

Una BD relacional consiste en una colección de tablas, a cada una de las cuales se les asigna un nombre único. Una fila de la tabla representa una relación entre un conjunto de valores, puesto que una tabla es una colección de dichas relaciones, hay una estrecha

correspondencia entre el concepto de tabla y el concepto matemático de relación, del cual toma su nombre el modelo de datos relacional.

Una base de datos relacional es una base de datos percibida por los usuarios como una colección de relaciones que varian en el tiempo, es decir, una colección de variables de relación.

Una vez diseñada la Base de Datos en el Modelo Entidad Relación, se procede a desarrollar el modelo en un ambiente físico, en el sistema desarrollado se utilizó el Modelo Relacional de Mysql.

2.2.2.1.1 Presentación y Objetivos.

La introducción realizada por Codd, muy a finales de los años sesenta, de la teoría matemática de las relaciones en el campo de las bases de datos supuso un importante paso en la investigación de los SGBD, suministrando un sólido fundamento teórico para el desarrollo, dentro de este enfoque relacional, de nuevos sistemas.

El documento de Codd propone un modelo de datos basado en la teoría de las relaciones, en donde los datos se estructuran lógicamente en forma de relaciones (tablas), siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras características de tipo físico. En palabras de CODD(1970), "la vista relacional de los datos... parece ser superior al modelo de grafos o en red... Proporciona un medio de describir datos con sus estructura natural únicamente, es decir, sin superponer ninguna estructura adicional con el propósito de su representación en la máquina".

El trabajo publicado por Codd en ACM presentaba un nuevo modelo de datos que perseguía una serie de objetivos, muchos de ellos comunes a tros modelos, que se pueden resumir en

los siguientes:

- Independencia física. El modo en que se almacenan los datos no debe influir en su manipulación lógica y, por tanto, los usuarios que acceden a esos datos no han de modificar sus programas por cambios en el almacenamiento físico.
- Independencia lógica. Añadir, eliminar o modificar cualquier elemento de la base de datos no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
- Flexibilidad. En el sentido de poder ofrecer a cada usuario los datos de la forma más adecuada a la correspondiente aplicación.
- Uniformidad. Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- Sencillez. Las características anteriores, así como unos lenguajes de usuarios muy sencillos, producen como resultado que el modelo de datos relacional sea fácil de comprender y de utilizar por parte del usuario final.

Para conseguir los objetivos citados, Codd introduce el concepto de relación (tabla), como estructura básica del modelo. Todos los datos de una base de datos se representan en forma de relaciones cuyo contenido varía en el tiempo. Una relación, en terminología relacional, es un conjunto de filas (tuplas) con unas determinadas características.

Con respecto a la parte dinámica del modelo, se propone un conjunto de operadores que se aplican a las relaciones. Algunos de estos operadores son clásicos de la teoría de conjuntos, mientras que otros fueron introducidos específicamente para el modelo relacional. Todos ellos conforman el álgebra relacional definida formalmente en CODD (1972), donde además

se compara el álgebra relacional con el cálculo relacional, otro lenguaje también propuesto por Codd EN este mismo trabajo.

Analizando la evolución del modelo relacional, se observa que después de los primeros estudios teóricos que se extienden a partir de 1970, comienza el desarrollo de diversos propósitos, entre los que destacan el Sistema R que se llevó a cabo en los laboratorios de IBM California y que fue el origen de los productos relacionales de IBM(DB2, SQL/DS, etc.), e INGRES, creando en la Universidad de Berkeley, STONEBRAKER (1986), que se transformó más tarde en un sistema comercial. A pesar de que desde su introducción en 1970, el modelo relacional se convirtió en uno de los principales temas de investigación en bases de datos, los primeros sistemas relacionales tardaron unos diez años en aparecer en el mercado, llegándose a calificar de juguetes, más aptos para la investigación o para el desarrollo de bases de datos experimentales o de pequeño tamaño que para el soporte de verdaderos sistemas de información. Los primeros productos comerciales basados en el modelo relacional fueron:

Query By Example –QBE-, (1978) un lenguaje de acceso relacional a ficheros VSAM de IBM, que no era, por tanto, un verdadero SGBDR.

Oracle(1979), primer SGBD relacional comercial, el cual soporta como lenguaje de identificación y manipulación de datos el SQL.

Ingres (1980), que tenia su origen en el prototipo de igual nombre de la universidad de Berkeley, y cuyo lenguaje basado en el cálculo relacional, el Quel, fue durante muchos años. Debido a sus características, el preferido por las universidads, especialmente en Estados Unidos.

A partir de 1980, un gran número de productos relacionales invaden el mercado, iniciándose un fuente debate entre los defensores del modelo relacional y sus detractores, estos últimos

ESTA TESIS NO SALE
DE LA BIBLIOTECA

consideraban que la falta de eficiencia de los productos relacionales impedían que fuesen una alternativa válida a los sistemas Jerárquicos y Codasyl imperantes en el momento.

2.2.2.1.2 Estructura del Modelo Relacional.

La relación es el elemento básico del modelo relacional, y se puede representar como una tabla, como lo presenta la siguiente imagen

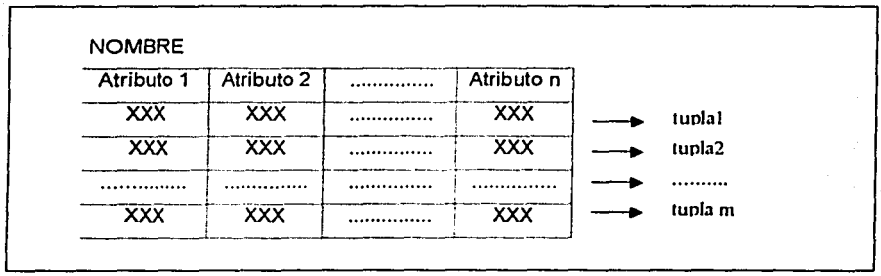


Imagen que representa una relación en forma de tabla

En ella podemos distinguir su nombre[5], un conjunto de columnas, denominadas atributos, que representan propiedades de la tabla y que también están caracterizadas por su nombre, y un conjunto de filas llamadas tuplas, contienen los valores que toma cada uno de los atributos para cada elemento de la relación. En la siguiente figura se representa la relación AUTOR, en donde aparece la estructura del modelo relacional. En ella podemos observar el nombre de la relación (AUTOR), los atributos (Nombre, Nacionalidad e Institución), los dominios (de donde los atributos tomas sus valores, varios atributos pueden tomar valores del mismo dominio), las tuplas (cada una de las cuales contiene los valores que toma el

[5] Puede haber relaciones, como aquellas que se obtienen como resultado de una consulta, que no tienen nombre.

nombre, la nacionalidad y la institución para un determinado autor), el grado (número de atributos), y la cardinalidad (número de tuplas).

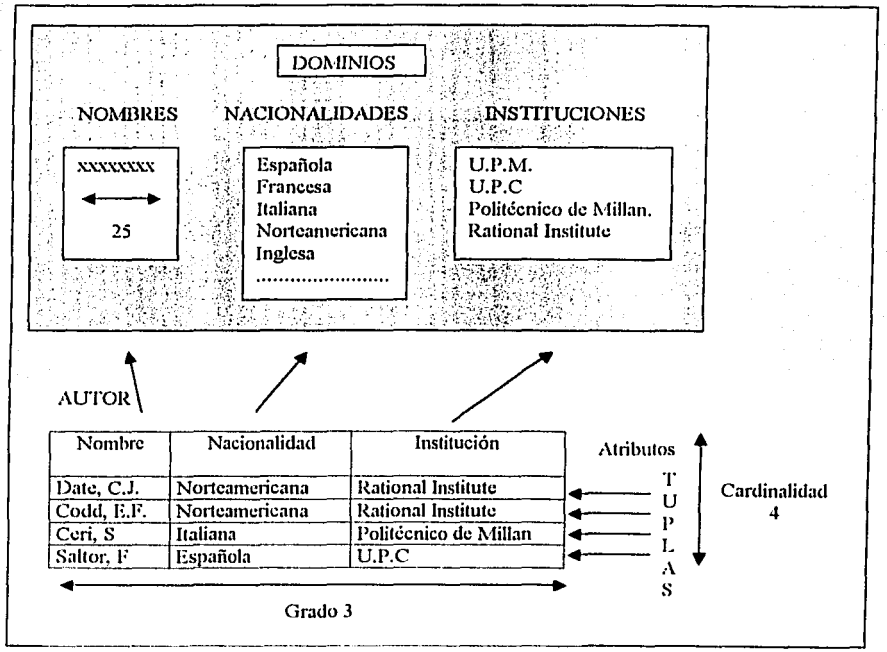


Imagen que representa la relación autor

Una relación se puede representar en forma de tabla, aunque tiene una serie de elementos característicos que la distinguen de la tabla, ya que no se admiten filas duplicadas, las filas y las columnas no están ordenadas y es plana, es decir, que en el cruce de una fila y de una columna sólo puede haber un valor (no se admiten atributos multivaluados).

En una tabla se pueden distinguir una cabecera que define la estructura de la tabla, es decir,

sus atributos como los dominios subyacentes, y un cuerpo que está formado por un conjunto de tuplas que varía en el tiempo.

Esta representación de la relación como una tabla ha sido el origen de que los productos relacionales y los usuarios utilicen habitualmente el nombre de tabla (en principio ajeno a la teoría relacional) para denominar las relaciones y, como consecuencia de ello, se llame filas a las tuplas y columnas a los atributos, si bien, la terminología.

En la siguiente tabla, se compara la terminología relacional con la que corresponde a las tablas y a los ficheros.

RELACION	TABLA	FICHERO
Tupla	Fila	Registro
Atributo	Columna	Campo
Grado	No. de Columnas	No. de Campo
Cardinalidad	No de Filas	No de Registro

2.2.2.2. Modelo de red.

Los datos en el modelo de red se representan mediante colecciones de registros y las relaciones entre los datos se representan mediante enlaces, los cuales pueden verse como punteros.

Los modelos de datos en red representan las entidades en forma de nodos de un grafo, y las asociaciones o interrelaciones entres éstas, mediante los arcos que unen dichos nodos. Esta representación, que no impone en principio ninguna restricción ni al tipo ni al número de los arcos, permite el modelado de estructuras de datos tan complejas como se desee.

2.2.2.2.1 Propuesta Codasyl

Un modelo de datos de tipo red que introduce determinadas restricciones inherentes es el denominado modelo Codasyl, que toma su nombre del comité que propuso sus especificaciones. Este modelo constituye una simplificación del modelo en red general en la que se admiten sólo determinados tipos de interrelaciones y se incluyen algunas restricciones adicionales, que, sin embargo, no limitan excesivamente la flexibilidad que proporciona el modelo en red general, facilitando en cambio una instrumentación eficiente.

2.2.2.2.2 Objetivos

El grupo Codasyl enunció una serie de objetivos que, en su opinión, debía perseguir cualquier cualquier sistema de gestión de bases de datos:

- Flexibilidad frente a los usuarios.
- Uso concurrente
- Diversas estratégicas de búsqueda.
- Seguridad.
- Gestión centralizada del almacenamiento físico
- Independencia del almacenamiento físico.
- Flexibilidad en el modelo de datos.
- Facilidad para el usuario
- Independencia de los programas respecto de los datos.
- Descripción de datos independiente de los programas.
- Independencia respecto a los lenguajes de programación.
- Interfaces con múltiples lenguajes.

2.2.2.3. Modelo jerárquico.

El modelo jerárquico es similar al modelo de red, los datos y las relaciones se representan mediante registros y enlaces. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles.

Entre los primeros modelos de datos que surgieron en los SGBD comerciales se encuentran las estructuras en árbol propias de los productos jerárquicos.

Los árboles, tan extendidos en informática, como instrumentos para la representación de estructuras de datos, presentan problemas por su poca flexibilidad lo que da origen a una falta de adaptación a muchas organizaciones reales.

Aunque no se ha llegado a una formalización matemática del modelo y de sus lenguajes, como ha ocurrido en el caso del relacional, ni tampoco se ha intentado su estandarización, como el Codasyl, sin embargo, los productos jerárquicos (el IMS y el DL/I de IBM como máximos exponentes de estos sistemas) consiguieron altas cuotas de mercado, aunque la actual difusión de la tecnología relacional los han llevado a convertirse en sistemas superados, lo cual no quiere decir que no persistan todavía importantes aplicaciones soportadas en estos productos que están trabajando, por su eficiente respuesta, a satisfacción de sus usuarios, siempre que las aplicaciones desarrolladas sobre ellos se mantengan sin apenas cambios.

La implementación del modelo Jerárquico en los productos se lleva a cabo en base a punteros: estructura física (organización y tipo de punteros) que varía según los productos, e incluso un mismo producto (por ejemplo, IMS) proporciona distintas organizaciones físicas a fin de que el usuario pueda conseguir una mayor eficiencia en el diseño físico de la base de datos.

2.2.2.3.1 Características de la estructura jerárquica

En el modelo de datos Jerárquico, el esquema es una estructura arborescente compuesta de nodos, que representan las entidades, enlazados por arcos, que representan las asociaciones o interrelaciones entre dichas entidades

La estructura del modelo de datos jerárquico es un caso particular de la del modelo en red, con fuertes restricciones adicionales derivadas de que las asociaciones del modelo jerárquico deben formar un árbol en el que el orden de los nodos es importante, tiene las siguientes características:

1. El árbol se organiza en un conjunto de niveles
2. El nodo raíz, el más alto de la jerarquía, se corresponde con el nivel 0 (algunos autores consideran que se trata del nivel 1).
3. Los arcos representan las asociaciones jerárquicas entre dos entidades y no tienen nombre, ya que no es necesario porque entre dos conjuntos de datos sólo puede haber una interrelación.
4. Mientras que un nodo de nivel superior (padre) puede tener un número ilimitado de nodos de nivel inferior (hijos), al nodo del nivel inferior sólo le puede corresponder un único nodo de nivel superior. En otras palabras, un progenitor (o padre) puede tener varios descendientes (o hijos), pero un hijo sólo tiene un padre.
5. Todo nodo, a excepción del nodo raíz, ha de tener obligatoriamente un padre.
6. Se llaman hojas los nodos que no tienen descendientes.
7. Se llama altura al número de niveles de la estructura jerárquica.
8. Se denomina momento al número de nodos.
9. Sólo están permitidas las interrelaciones 1:1 ó 1:N.
10. Cada nodo no terminal y sus descendientes forman un subárbol, de forma que un

árbol es una estructura recursiva.

2.2.2.4. Diferencias entre modelos.

Los modelos relacionales se diferencian de los modelos de red y jerárquico en que no usan punteros o enlaces. En cambio, el modelo relacional conecta registros mediante los valores que éstos contienen. Esta libertad del uso de punteros permite que se defina una base matemática formal.

2.2.3. Modelos físicos de datos.

Los modelos físicos de datos se usan para describir datos en el nivel más bajo. Hay muy pocos de modelos físicos de datos en uso, siendo los más conocidos el modelo unificador y de memoria de elementos.

2.2.4. Los modelos de datos en el diseño de bases de datos

Los modelos de datos son un eficaz instrumento en el diseño de bases de datos. Los niveles de abstracción de la arquitectura ANSI facilitan el diseño de una base de datos, al proporcionar nuevos instrumentos que ayudan a la estructuración, paso a paso, del mundo real hasta llegar a la base de datos física.

En el estado actual de la técnica es conveniente, en el diseño de bases de datos, distinguir la fase de modelado conceptual, que es la descripción del mundo real (empresa o administración) de acuerdo con un modelo altamente semántico e independiente del SGBD en el que posteriormente se vaya a hacer la implementación de la base de datos, y la fase de diseño lógico, en la cual se ha de obtener un esquema que responda a la estructura lógica específica (en general, relacional) del SGBD que se aplique en cada caso, por lo que dicho

esquema está sometido a las restricciones que imponga el modelo del SGBD en concreto.

En un primer paso, con la ayuda del modelo conceptual, se obtiene el esquema conceptual, aunque a veces no se formaliza este paso y el analista, sin una metodología precisa, hace una abstracción del mundo real que es lo que hemos llamado estructura percibida. A continuación, aplicando al esquema conceptual las reglas del modelo de datos propio del SGBD que se va a utilizar, se obtiene el esquema lógico (también llamado esquema de base de datos), de éste se pasa al esquema interno, donde el objetivo es conseguir la máxima eficiencia de cara a la máquina y al problema específico. Por último, se implementa la base de datos física en los soportes secundarios. La estructura física se ha de rellenar con los valores (ocurrencias) que se obtienen por observación de los sucesos del mundo real.

En el enfoque que acabamos de presentar nos estamos moviendo en tres mundos distintos, el de la realidad, el de las ideas y el de los datos, mundos que se confunden a menudo, de forma que todos pasamos a veces de uno a otro sin advertencia previa.

En el primero de estos dominios, el mundo real (al que también llamamos Universo del Discurso), existen objetos y asociaciones entre ellos, ambos tienen propiedades y hay reglas que imponen ciertas limitaciones. Será necesaria una abstracción de este mundo real por parte de los diseñadores de la base de datos que lo observa a fin de obtener un esquema conceptual, que será tanto más perfecto cuanto más se asemeje al mundo real que se está contemplando bajo la perspectiva de unos determinados objetivos (los que nos impone nuestro sistema de información en concreto), éste es el dominio de las ideas y de la información. En el modelo Entidad / Relación (así como en otros modelos conceptuales), el acento se pone en este campo, en el que aparecen entidades e interrelaciones con determinados atributos, así como restricciones semánticas. Hay que llamar la atención sobre el hecho de que la mayor parte de los modelos actuales no son los suficientemente ricos

desde el punto de vista semántico y no proporcionan los deseables instrumentos para captar el contenido semántico del mundo real.

Al no comprender el SGBD los conceptos del esquema conceptual, es preciso pasar a una descripción en los términos propios del SGBD, para llegar por último al mundo de los datos en el cual éstos se almacenan en la estructura física previamente descrita, donde tendremos cadena de bits, totalmente carentes de significado si no disponemos de los medios que nos permitan recorrer el camino inverso pasando de nuevo al mundo real con ayuda del lenguaje de manipulación, por medio del cual actualizaremos o recuperaremos los datos almacenados en la base, reincorporándoles su contenido semántico y obteniendo la información que necesita el usuario.

Las herramientas CASE proporcionan una importante ayuda en el diseño de bases de datos, al disponer de modelos de datos semánticos (en general basados en el modelo E/R, con algunas extensiones) que facilitan el diseño conceptual y realizan la transformación al modelo relacional propio de los productos comerciales más extendidos.

2.3 Seguridad de Bases de Datos

El aspecto global de seguridad de los datos está muy vinculado al propio concepto de lo que es la base de datos: "Un conjunto de datos integrados, adecuado a varios usuarios y aplicaciones". Es el propio uso concurrente de los datos el que, en muchos casos, plantea problemas de seguridad que el administrador de la base de datos debe aminorar, en la medida de lo posible, con las facilidades que le proporciona el SGBD.

La protección de los datos deberá llevarse a cabo contra fallos físicos (de unidad central de proceso, de memoria secundaria, de memoria principal, etc.) y fallos humanos, ya sean éstos intencionados o no. Estos fallos alteran indebidamente los datos, los corrompen, con lo que

la base de datos ya no puede servir a los fines para los que fue creada.

El SGBD facilita normalmente mecanismos para prevenir los fallos (subsistema de control), para detectarlos una vez que se han producido (subsistema de detección) y para corregirlos después de haber sido detectados (subsistemas de recuperación).

Actualmente se considera generalmente aceptado que la seguridad comprende tres aspectos fundamentales:

- Confidencialidad, es decir, no desvelar datos a usuarios no autorizados, que comprende también la privacidad (la protección de datos personales).
- Accesibilidad, esto es, que la información se encuentre disponible.
- Integridad, que permite asegurar que los datos no se han falseado[6].

Hay que tener en cuenta que tanto las amenazas como los mecanismos para contrarrestarlas, suelen afectar a estas tres características de forma conjunta. Así, por ejemplo, fallos del sistema que hacen que la información no sea accesible pueden llevar consigo una pérdida de integridad.

También debemos ser consientes de que las medidas de seguridad que deberían establecerse en cuanto a los datos, no sólo afectan al SGBD, sino que comprenden también el hardware y el sistema operativo, las medidas de seguridad física, los controles organizativos (políticas, normas, procedimientos, etc.) y legales.

[6] CASTAÑO ADORACION DE MIGUEL, PIATTINI MARIO.: Fundamentos y Modelos de Bases de Datos, México, 1993, Alfaomega, 2ª Edición, pág. 421

3.HERRAMIENTAS DE LIBRE DISTRIBUCION

3.1 ¿QUÉ ES EL SOFTWARE LIBRE?

"Software Libre" es cuestión de libertad, no de precio. "Free" en "free software" es una palabra que debe ser traducida como "libre" tal como en "libertad de expresión" ("Free speech"); no como "Gratis" como en "cerveza gratuita" ("Free beer").

"Software Libre" se refiere a la libertad de los usuarios de correr, copiar, distribuir, estudiar, cambiar y mejorar el software. Mas precisamente, se refiere a las cuatro libertades de los usuarios de software[1]:

a) La libertad de correr el programa, con cualquier propósito (libertad 0).

b) La libertad de estudiar como funciona el programa, y adaptarlo a sus necesidades (libertad 1). El acceso al código fuente es una precondición para esto.

c) La libertad de distribuir copias de manera que se puede ayudar al vecino (libertad 2).

d) La libertad de mejorar el programa, y liberar las mejoras al publico de tal manera que toda la comunidad se beneficia. (libertad 3). El acceso al código fuente es una precondición para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Por esto, eres libre de redistribuir copias, ya sea con o sin modificaciones, ya sea gratis o cobrando una cuota por la distribución a cualquiera y a cualquier lugar. El ser libre de hacer esto significa (entre otras cosas) que no tienes que pedir o pagar permisos.

También se tiene la libertad de hacer modificaciones y utilizarlas de manera privada para el trabajo o juego, sin ni siquiera mencionar que dichas modificaciones existen. Si publica sus

---

[1] Para mayor referencia consultar la dirección http://www.gnu.org/philosophy/free-sw.es.html

cambios, no necesita avisar a nadie en particular, o de una manera en particular.

Se pudo haber pagado por obtener copias del software, o tal vez se obtuvieron las copias sin ningún cargo. Pero no importa como se obtuvieron las copias, siempre se tiene la libertad de copiar y cambiar el software.

Para que estas libertades sean reales, deben ser irrevocables conforme no se haga algo malo, el desarrollador del software tiene el poder de revocar la licencia, inclusive si no se ha dado causa, el software no es libre.

Sin embargo, algunos tipos de reglas acerca de la manera de distribuir software libre son aceptables, cuando no entran en conflicto con las libertades centrales. Por ejemplo, copyleft (expresado muy simplemente) es la regla que implica que cuando se redistribuya el programa, no se puede agregar restricciones para denegar a otras personas las libertades centrales. Esta regla no entra en conflicto con las libertades centrales; sino que las protege.

Las reglas acerca de como empacar una versión modificada son aceptables, si ellas no bloquean efectivamente la libertad de liberar las versiones modificadas. Reglas que indican "Si tu haces disponible el programa de esta manera, debes también hacerlo disponible en esta otra" pueden ser aceptables también, bajo la misma condición. (Nótese que tal regla deja la elección de hacer o no disponible el programa).

Algunas veces las reglas de control de exportación del gobierno y las sanciones del mercado pueden reprimir la libertad de distribuir copias de los programas a nivel internacional. Los desarrolladores de software no tienen el poder para eliminar o sobrepasar estas restricciones, pero lo que pueden y deben hacer es rehusar el imponerlas como condiciones de uso del programa. De esta manera, las restricciones no afectarán actividades y gente fuera de las jurisdicciones de estos gobiernos.

Cuando se habla de software libre, es mejor evitar términos como: "regalar", o "gratis",

porque esos términos implican que lo importante es el precio, y no la libertad. Algunos términos comunes tales como "piratajе" contraen opiniones que esperamos no se apoye.

3.1.1 Como obtener software libre

El software libre, está disponible de varias formas:

- Comprar copias en CD-ROM de la Fundación para Software Libre [FSL]
- Bajarlo por FTP(File Transfer Protocol).
- Obtener una copia de un amigo.
- Comprar una computadora con un sistema GNU/Linux completamente libre preinstalado de una de las compañías que pueden ofrecerlos.

3.1.2 Categorías del sofware libre y no libre

He aquí un glosario de varias categorías de software que son mencionadas frecuentemente en discusiones de software libre. Explica qué categorías se traslapan o son parte de otras categorías[2].

1. Software libre. El software libre es software que viene con autorización para que cualquiera pueda usarlo, copiarlo y distribuirlo, ya sea literal o con modificaciones, gratis o mediante una gratificación.

Esto significa que el código fuente debe estar disponible. ``Si no es fuente, no es software".

Si un programa es libre, entonces puede ser potencialmente incluido en un sistema operativo libre tal como GNU, o sistemas GNU/Linux libres.

Hay muchas maneras diferentes de hacer un programa libre. Algunas de las posibles

[2] Para mayor referencia consultar la dirección http://www.gnu.org/philosophy/categories.es.html

variaciones serán descritas posteriormente.

El software libre es frecuentemente más fiable que el software no libre.

2. Software de Fuente Abierta . El término software de ``fuente abierta" es usado por algunas personas para dar a entender más o menos lo mismo que software libre. Nosotros preferimos el término "software libre".

3. Software de dominio público. El software de dominio público es software que no está protegido con copyright. Es un caso especial de software libre no protegido con copyleft, que significa que algunas copias o versiones modificadas no pueden ser libres completamente.

Algunas veces la gente utiliza el término "dominio público" de una manera imprecisa para decir "libre" o "disponible gratis." Sin embargo, "dominio público" es un término legal y significa de manera precisa "sin copyright". Por claridad, recomendamos el uso de "dominio público" para ese significado solamente y el uso de otros términos para transmitir los otros significados.

4. Software protegido con copyleft. El software protegido con copyleft es software libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando éstos redistribuyen o modifican el software. Esto significa que cada copia del software, aun si ha sido modificado, debe ser software libre.

En el Proyecto GNU, se protegen mediante copyleft casi todo el software que se desarrolla, porque el objetivo es dar a cada usuario las libertades que el término ``software libre"implica.

Copyleft es un concepto general; para proteger actualmente un programa con copyleft, necesita usar un conjunto específico de términos de distribución. Hay muchas maneras posibles de escribir términos copyleft de distribución.

5. Software libre no protegido con copyleft. El software libre no protegido con copyleft viene desde el autor con autorización para redistribuir y modificar así como para añadirle

restricciones adicionales.

Si un programa es libre pero no protegido con copyleft, entonces algunas copias o versiones modificadas pueden no ser libres completamente. Una compañía de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como un producto propietario de software.

El Sistema X Window ilustra esto. El Consorcio X libera X11 con términos de distribución que lo hacen software libre no protegido con copyleft. Si se desea, se puede obtener una copia que tenga esos términos de distribución y es libre. Sin embargo, hay versiones no libres también, y hay estaciones de trabajo populares y tarjetas gráficas para PC para las cuales versiones no libres son las únicas que funcionan. Si se está usando este hardware, X11 no es software libre.

6. Software abarcado por GPL. La GPL (General Public License/Licencia Pública General) de GNU es un conjunto específico de términos de distribución para proteger con copyleft a un programa. El Proyecto GNU la utiliza como los términos de distribución para la mayoría del software GNU.

7. El sistema GNU. El sistema GNU es un sistema operativo libre completo estilo Unix.

Un sistema operativo libre estilo Unix consiste en muchos programas. El sistema GNU incluye todo el software GNU, así como muchos otros paquetes tales como el Sistema X Window y TeX que no son software GNU.

Debido a que el propósito de GNU es ser libre, cada componente individual en el sistema GNU tiene que ser software libre. No todos tienen que estar protegidos con copyleft, sin embargo; cualquier tipo de software libre es legalmente apto de incluirse si ayuda a alcanzar metas técnicas. Podemos hacer uso de software libre no protegido con copyleft como el Sistema X Window.

8. Software GNU. Software GNU es software que es liberado bajo el auspicio del Proyecto GNU. La mayoría del software GNU está protegido con copyleft, pero no todos; sin embargo, todo el software GNU debe ser software libre.

Algo de software GNU es escrito por el personal de la Fundación para el Software Libre (Free Software Foundation), pero la mayoría del software GNU es aportada por voluntarios. Parte del software aportado está protegido con copyright por la Fundación para el Software Libre; otra parte está protegido con copyright por los aportadores que los escribieron.

9. Software semilibre . El software semilibre es software que no es libre, pero viene con autorización para particulares de usar, copiar, distribuir y modificar (incluyendo la distribución de versiones modificadas) sin fines de lucro. PGP es un ejemplo de un programa semilibre.

El software semilibre es mucho mejor que el software propietario, pero aún plantea problemas y no podemos usarlo en un sistema operativo libre.

Las restricciones del copyleft están diseñadas para proteger las libertades esenciales de todos los usuarios. Para nosotros, la única justificación para cualquier restricción substantiva acerca del uso de un programa es prevenir la adición de restricciones por parte de otras personas. Los programas semilibres tienen restricciones adicionales, motivados por fines puramente egoístas.

Es imposible incluir software semilibre en un sistema operativo libre. Esto obedece a que los términos de distribución para el sistema operativo libre como un todo es la conjunción de los términos de distribución de todos los programas en él. Agregando un programa semilibre al sistema haría al sistema como un todo sólo semilibre. Existen dos razones por las que no queremos que esto suceda:

Creemos que el software libre debe ser para todos--incluyendo empresas, no solamente para

escuelas o para quienes gusten de matar el tiempo.

La distribución comercial de sistemas operativos libres, incluyendo sistemas GNU basados en Linux, es muy importante y los usuarios aprecian poder comprar distribuciones comerciales en CD-ROM. Incluyendo un programa semilibre en un sistema operativo cortaría su distribución comercial en CD-ROM.

La misma Fundación para el Software Libre es no comercial y por lo tanto estaríamos habilitados legalmente para usar un programa semilibre "internamente".

10. Software propietario. El software propietario es software que no es libre ni semilibre. Su uso, redistribución o modificación está prohibida, o requiere que se solicite autorización o está tan restringida que no pueda hacerla libre de un modo efectivo.

La Fundación para el Software Libre sigue la regla de no instalar ningún programa propietario en nuestras computadoras excepto temporalmente para el propósito específico de escribir un reemplazo libre para ese programa. Con excepción de lo anterior, sentimos que no hay excusa posible para instalar un programa propietario.

11. Freeware. El término ``freeware" no tiene una definición clara aceptada, pero es usada comúnmente para paquetes que permiten la redistribución pero no la modificación (y su código fuente no está disponible). Estos paquetes no son software libre.

12. Shareware. El shareware es software que viene con autorización para la gente de redistribuir copias, pero dice que quien continúe haciendo uso de una copia deberá pagar un cargo por licencia.

El shareware no es software libre, ni siquiera semilibre. Existen dos razones por las que no lo es: Para la mayoría del shareware, el código fuente no está disponible; de esta manera, no puede modificar el programa en absoluto.

El shareware no viene con autorización para hacer una copia e instalarlo sin pagar una

cantidad por licencia, ni aún para particulares involucrados en actividades sin ánimo de lucro. (En la práctica, la gente a menudo hace caso omiso a los términos de distribución y lo hace de todas formas, pero los términos no lo permiten.)

13. Software Comercial. El software comercial es software que está siendo desarrollado por una entidad que tiene la intención de hacer dinero del uso del software. "Comercial" y "propietario" . La mayoría del software comercial es propietario, pero hay software libre comercial y hay software no libre no comercial.

Por ejemplo, Ada de GNU siempre es distribuida bajo los términos de la GPL de GNU y cada copia es software libre; pero los desarrolladores venden contratos de soporte. Cuando sus vendedores les hablan a los clientes potenciales, algunas veces el cliente dice "Nos sentiríamos más seguros con un compilador comercial". Los vendedores responden, "Ada de GNU es un compilador comercial; sólo que es software libre."

Para el proyecto GNU, el énfasis está en otro orden: lo importante es que Ada de GNU es softwar libre; si es comercial no es una pregunta importante. Sin embargo, el desarrollo adicional de Ada de GNU que resulta del negocio que soporta es definitivamente beneficioso.

3.2 SISTEMA OPERATIVO LINUX

3.2.1 Introducción

La mayor parte de los ordenadores que existen en la actualidad están diseñados de forma que puedan ejecutar diversas tareas o programas. Estos programas pueden ir desde un procesador de textos, a un programa para la animación de gráficos tridimensionales o distintos tipos de juegos. Para su correcto funcionamiento deben ser además capaces de acceder a las recursos de que dispone el ordenador, como por ejemplo escribir o leer datos en un disco duro, mostrar un gráfico por pantalla, etc. Es evidente, que si cada programa

actuase de una forma independiente, existirían graves problemas y conflictos, puesto que, por ejemplo, tendrían libertad para escribir sus datos sobre los de otro, etc.

Para solucionar este tipo de problemas se desarrollaron los Sistemas Operativos, los cuales aportan unos mecanismos y reglas básicas de funcionamiento, de forma que los programas puedan acceder a los recursos del ordenador de una forma adecuada. Aunque ésta fue la funcionalidad inicial de los sistemas operativos, con el tiempo se han añadido otras muchas, como la ejecución de programas, el control de la memoria del ordenador, la creación y control de interfaces gráficas de usuario, etc.

En la actualidad existen una gran cantidad de sistemas operativos dependiendo del tipo de ordenador en el que se va a ejecutar. Por ejemplo para los PC uno de los sistemas operativos más difundidos en Microsoft Windows, tanto en las versiones 95, 98, 2000, NT 4.0 etc. Otros posibles sistemas operativos para este tipo de ordenadores son Solaris, OS/2, BeOS, Microsoft DOS, o uno de los sistemas operativos más poderosos y en rápida expansión para PC, Linux.

3.2.2 Historia de Linux

UNIX es uno de los sistemas operativos más populares del mundo debido a su extenso soporte y distribución. Originalmente fue desarrollado como sistema multitarea con tiempo compartido para miniordenadores y mainframes a mediados de los 70, y desde entonces se ha convertido en uno de los sistemas más utilizados a pesar de su, ocasionalmente, confusa interfaz con el usuario y el problema de su estandarización.

¿Cuál es la verdadera razón de la popularidad de UNIX? Muchos hackers consideran que UNIX es el auténtico y único sistema operativo. El desarrollo de Linux parte de un grupo en expansión de hackers de UNIX que quisieron hacer su sistema con sus propias manos.

Existen numerosas versiones de UNIX para muchos sistemas, desde ordenadores personales hasta supercomputadoras.

Linux es una versión de UNIX de libre distribución, inicialmente desarrollada por Linus Torvalds en la Universidad de Helsinki, en Finlandia. Fue desarrollado con la ayuda de muchos programadores y expertos de Unix a lo largo y ancho del mundo, gracias a la presencia de Internet. Cualquier habitante del planeta puede acceder a Linux y desarrollar nuevos módulos o cambiarlo a su antojo. El núcleo de Linux no utiliza ni una sola línea del código de AT&T o de cualquier otra fuente de propiedad comercial, y buena parte del software para Linux se desarrolla bajo las reglas del proyecto de GNU de la Free Software Foundation, Cambridge, Massachusetts.

Inicialmente, sólo fue un proyecto de aficionado de Linus Torvalds. Se inspiraba en Minix, un pequeño UNIX desarrollado por Andy Tanenbaum, y las primeras discusiones sobre Linux surgieron en el grupo de News comp.os.minix. Estas discusiones giraban en torno al desarrollo de un pequeño sistema UNIX de carácter académico dirigido a aquellos usuarios de Minix que querían algo más.

El desarrollo inicial de Linux ya aprovechaba las características de conmutación de tareas en modo protegido del 386, y se escribió todo en ensamblador, al final de Agosto de 1991 Linus tenía ya la versión 0.01, un driver de discos con numerosos bugs, así como un pequeño sistema de ficheros pero que parecían funcionar en una PC.

No se anunció nada sobre esa versión, puesto que las fuentes del 0.01 jamás fueron ejecutables: contenían solo rudimentos de lo que sería el núcleo, y se asumía que se tenía acceso a un Minix para poderlo compilar y jugar con él.

El 5 de Octubre de 1991, Linus anunció la primera versión "oficial" de Linux, la 0.02. Ya podía ejecutar bash (el shell de GNU) y gcc (el compilador de C de GNU), pero no hacía mucho

más. La intención era ser un juguete para hackers. No habla nada sobre soporte a usuarios, distribuciones, documentación ni nada parecido.

Tras la versión 0.03, Linus saltó a la versión 0.10, al tiempo que más gente empezaba a participar en su desarrollo. Tras numerosas revisiones, se alcanzó la versión 0.95, reflejando la esperanza de tener lista muy pronto una versión "oficial". (Generalmente, la versión 1.0 de los programas se corresponden con la primera teóricamente completa y sin errores). Esto sucedía en Marzo de 1992. Año y medio después, en Diciembre del 93, el núcleo estaba en la revisión 0.99.pl14, en una aproximación asintótica al 1.0.

Hoy Linux es ya un clónico de UNIX completo, capaz de ejecutar X Window, TCP/IP, Emacs, UUCP y software de correo y News. Mucho software de libre distribución ha sido ya portado a Linux, y están empezando a aparecer aplicaciones comerciales. El hardware soportado es mucho mayor que en las primeras versiones del núcleo.

3.2.3 ¿Qué son las distribuciones?

Uno de los primeros conceptos que aparecen al iniciarse en Linux es el concepto de distribución. Una distribución es un agrupamiento del núcleo del sistema operativo Linux (la parte desarrollada por Linus Torvalds) y otra serie de aplicaciones de uso general o no tan general.

En principio las empresas que desarrollan las distribuciones de Linux están en su derecho al cobrar una cierta cantidad por el software que ofrecen, aunque en la mayor parte de las ocasiones se pueden conseguir estas distribuciones desde Internet, de revistas o de amigos, siendo todas estas formas gratuitas y legales.

Las distribuciones más conocidas son RedHat, Debian, Slackware, SuSE, Corel Linux y Mandrake todas ellas incluyen el software más reciente y empleado lo cual incluye

compiladores de C/C++, editores de texto, juegos, programas para el acceso a Internet, así como el entorno gráfico de Linux: X Window.

Para el desarrollo del sistema, se manejó la distribución de Linux Mandrake 8.0.

3.2.4 Características del sistema

Linux implementa la mayor parte de las características que se encuentran en otras implementaciones de UNIX, más algunas otras que no son habituales

Linux es un sistema operativo completo con multitarea y multiusuario (como cualquier otra versión de UNIX). Esto significa que pueden trabajar varios usuarios simultáneamente en él, y que cada uno de ellos puede tener varios programas en ejecución.

El sistema Linux es compatible con ciertos estándares de UNIX a nivel de código fuente, incluyendo el IEEE.POSIX.1, System V y BSD. Fue desarrollado buscando la portabilidad de los fuentes: encontrará que casi todo el software gratuito desarrollado para UNIX se compila en Linux sin problemas. Y todo lo que se hace para Linux (código del núcleo, drivers, librerías y programas de usuario) es de libre distribución.

En Linux también se implementa el control de trabajos POSIX (que se usa en los shells csh y bash), las pseudoterminales (dispositivos pty), y teclados nacionales mediante manejadores de teclado cargables dinámicamente. Además, soporta consolas virtuales, lo que permite tener más de una sesión abierta en la consola de texto y conmutar entre ellas fácilmente.

Linux soporta diversos sistemas de ficheros para guardar los datos. Algunos de ellos, como el ext2fs, han sido desarrollados especificamente para Linux. Otros sistemas de ficheros, como el Minix-1 o el de Xenix también están soportados. Y con el de MS-DOS se podrán acceder desde Linux a los disquetes y particiones en discos duros formateados con MS-DOS. Además, también soporta el ISO-9660, que es el estándar seguido en el formato de los

CD-ROMs.

Linux implementa todo lo necesario para trabajar en red con TCP/IP. Desde manejadores para las tarjetas de red más populares hasta SLIP/PPP, que permiten acceder a una red TCP/IP por el puerto serie. También se implementan PLIP (para comunicarse por el puerto de la impresora) y NFS (para acceso remoto a ficheros). Y también se han portado los clientes de TCP/IP, como FTP, telnet, NNTP y SMTP.

El núcleo de Linux ha sido desarrollado para utilizar las características del modo protegido de los microprocesadores 80386 y 80486. En concreto, hace uso de la gestión de memoria avanzada del modo protegido y otras características avanzadas.

El núcleo soporta ejecutables con paginación por demanda. Esto significa que sólo los segmentos del programa que se necesitan se cargan en memoria desde el disco. Las páginas de los ejecutables son compartidas mediante la técnica copy-on-write, contribuyendo todo ello a reducir la cantidad de memoria requerida para las aplicaciones.

Con el fin de incrementar la memoria disponible, Linux implementa la paginación con el disco: puede tener hasta 256 megabytes de espacio de intercambio o "swap" (3) en el disco duro. Cuando el sistema necesita más memoria, expulsará páginas inactivas al disco, permitiendo la ejecución de programas más grandes o aumentando el número de usuarios que puede atender a la vez. Sin embargo, el espacio de intercambio no puede suplir totalmente a la memoria RAM, ya que el primero es mucho más lento que ésta.

La memoria dedicada a los programas y a la cache de disco está unificada. Por ello, si en cierto momento hay mucha memoria libre, el tamaño de la cache de disco aumentará acelerando así los accesos.

Los ejecutables hacen uso de las librerías de enlace dinámico. Esto significa que los ejecutables comparten el código común de las librerías en un único fichero, como sucede en

SunOS. Así, los ejecutables serán más cortos a la hora de guardarlos en el disco, incluyendo aquellos que hagan uso de muchas funciones de librería. También pueden enlazarse estáticamente cuando se deseen ejecutables que no requieran la presencia de las librerías dinámicas en el sistema. El enlace dinámico se hace en tiempo de ejecución, con lo que el programador puede cambiar las librerías sin necesidad de recompilación de los ejecutables. Para facilitar la depuración de los programas, el núcleo de Linux puede generar volcados de la imagen de memoria de los programas (ficheros core). Entre esto y la posibilidad de compilar ejecutables con soporte de depuración, el programador podrá averiguar la causa de los fallos de su programa[3].

3.3 BASE DE DATOS MySQL

3.3.1 ¿Qué es SQL?

SQL es un método basado en un potente lenguaje, para organizar, administrar y consultar datos almacenados en una computadora. SQL es una sigla que deviene de su nombre en inglés "Structured Query Language" (Lenguaje de Consulta Estructurado). Más específicamente SQL esta definido en torno al modelo de bases de datos relacionales, basado en el álgebra relacional, esto le da a SQL las ventajas que lo imponen como el sistema de mayor aceptación. Algunas de las ventajas son:

- Marco teórico sólido, fundamentado en el álgebra relacional
- Simplicidad de conceptos (modelo de base de datos: tablas=líneas x columnas)
- Definición de vínculos en la consulta, esto le da a SQL una gran flexibilidad
- Fácil y rápido aprendizaje

[3] Para mayor referencia consultar la dirección http://www.linux-mandrake.com/en/ y http://www.linux.com

•Arquitectura cliente-servidor

•Integración con cualquier lenguaje de programación

•Estandarización

3.3.2 Historia de MySQL

IBM empezó a comercializar en 1981 el SQL y desde entonces este producto ha tenido un papel importante en el desarrollo de la bases de datos relacionales. IBM propuso y fue aceptada , una versión de SQL al Instituto de Estándares Nacional Americano(ANSI) y desde entonces es utilizado de forma generalizada en las bases de datos relacionales. En 1983 nació DB2 la más popular(por lo menos en los grandes ordenadores) de las bases de datos de este tipo hasta estos mismos momentos.

Alrededor de la década del 90, Michael Windenis (monty@analytikerna.se) comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel (ISAM). Sin embargo, después de algunos testeos llego a la conclusión que mSQL no era lo suficientemente rápido ni flexible para sus necesidades. De todo esto surgió en una nueva interfaz SQL (claro que con código mas portable) con algunas apariencias similares en la API de C y en los nombres y funciones de muchos de sus programas.

Esto habla sido hecho para lograr con relativa facilidad portar aplicaciones y utilidades de MiniSQL a MySQL.

El Origen del nombre MySQL no esta perfectamente claro. Algunos lo atribuyen al hecho de que un gran numero de nuestras librerias y herramientas le asignamos el prefijo "My" por costumbre. Sin embargo la hija de 'Monty' es ademas llamada My. Asi que cual de las dos razones da el nombre a MySQL es aun un misterio.

3.3.3 ¿Que es MySQL ?

MySQL es un Sistema de Gestión de Base de Datos.

Una Base de Datos es una colección estructurada de datos. Puede ser, desde una simple lista de artículos a las inmensas cantidades de información en una red corporativa.

MySQL es un gestor de bases de datos SQL (Structured Query Language). Es una implementación Cliente-Servidor que consta de un servidor y diferentes clientes (programas / librerías). Podemos agregar, acceder, y procesar datos grabados en una base de datos. Actualmente el gestor de base de datos juega un rol central en la informática, como única utilidad, o como parte de otra aplicación.

Es un Sistema de Gestión de Base de Datos Relacional. El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos. Esto añade velocidad y flexibilidad.

MySQL es un software de código abierto esto quiere decir que es accesible para cualquiera, para usarlo o modificarlo. Podemos descargar MySQL desde Internet y usarlo sin pagar nada, de esta manera cualquiera puede inclinarse a estudiar el código fuente y cambiarlo para adecuarlo a sus necesidades. MySQL usa el GPL (GNU Licencia Publica General) para definir que podemos y no podemos hacer con el software en diferentes situaciones. Entre otras cuestiones esta licencia aclara que no cuesta dinero a menos que lo incluyamos en un software comercial y tenemos el código fuente.

3.3.4 ¿Porque usar MySQL?

MySQL es muy rápido, confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños (siempre, claro está, comparada con las de su categoría). Además

tiene un conjunto muy practico de características desarrolladas en cooperación muy cercana con los usuarios.

Sin embargo bajo constante desarrollo, MySQL hoy en día ofrece un rico y muy útil conjunto de funciones. La conectividad, velocidad y seguridad hace de MySQL altamente conveniente para acceder a bases de datos en Internet.

En el proyecto de tesis se utilizó la versión 3.23.41.

3.3.5 Principales Características

- El principal objetivo de MySQL es velocidad y robustez.
- Escrito en C y C++, testado con GCC 2.7.2.1. Usa GNU autoconf para potabilidad.
- Clientes C, C++, JAVA, Perl, TCL, PHP.
- Multiproceso, es decir puede usar varias CPU si éstas están disponibles.
- Puede trabajar en distintas plataformas y S.O. distintos.
- Sistema de contraseñas y privilegios muy flexible y segura.
- Todas la palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- 16 Indices por tabla, cada Indice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.
- Todas las columnas pueden tener valores por defecto.
- Utilidad (Isamchk) para chequear, optimizar y reparar tablas.
- Todos los datos están grabados en formato ISO8859_1.
- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.
- Todos los comandos tienen -help o -? Para las ayudas.

- Diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.
- ODBC para Windows 95 (con fuentes), se puede utilizar ACCESS para conectar con el servidor[4].

3.4 LENGUAJES DE PROGRAMACION

3.4.1 historia de PHP

PHP fue concebido en otoño de 1994 por Rasmus Lerdorf. Las primeras versiones no distribuidas al público fueron usadas en sus páginas web para mantener un control sobre quien consultaba su curriculum. La primera versión disponible para el público a principios de 1995 fue conocida como "Herramientas para paginas web personales" (Personal Home Page Tools). Consistían en un analizador sintáctico muy simple que solo entendía unas cuantas macros y una serie de utilidades comunes en las páginas web de entonces, un libro de visitas, un contador y otras pequeñas cosas.

El analizador sintáctico fue reescrito a mediados de 1995 y fue nombrado PHP/FI versión 2. FI viene de otro programa que Rasmus había escrito y que procesaba los datos de formularios. Así que combinó las "Herramientas para paginas web personales", el "intérprete de formularios", añadió soporte para mSQL y PHP/FI vio la luz. PHP/FI creció a gran velocidad y la gente empezó a contribuir en el código.

Es difícil dar estadísticas exactas, pero se estima que a finales de 1996 PHP/FI se estaba usando al menos en 15.000 páginas web alrededor del mundo.

A mediados de 1997 este número había crecido a mas de 50.000. A mediados de 1997 el

[4] Para mayor referencia consultar la dirección http://www.mysql.com

desarrollo del proyecto sufrió un profundo cambio, dejó de ser un proyecto personal de Rasmus, al cual habían ayudado un grupo de usuarios y se convirtió en un proyecto de grupo mucho más organizado. El analizador sintáctico se reescribió desde el principio por Zeev Suraski y Andi Gutmans y este nuevo analizador estableció las bases para PHP versión 3. Gran cantidad de código de PHP/FI fue portado a PHP3 y otra gran cantidad fue escrito completamente de nuevo.

A finales 1999, tanto PHP/FI como PHP3 se distribuyeron en un gran número de productos comerciales tales como el servidor web "C2's StrongHold" y Redhat Linux. Una estimación conservativa basada en estadísticas de NetCraft, es que más de 1.000.000 de servidores alrededor del mundo usan PHP.

Para hacernos una idea, este número es mayor que el número de servidores que utilizan el "Netscape's Enterprise server" en Internet. El trabajo de la actual generación de PHP, utiliza el potente motor de scripts Zend para proporcionar altas prestaciones, así como soporta otros servidores web, además de apache, que corren PHP como módulo nativo.

3.4.2 Qué es PHP?

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

Php no es lo mismo que un script CGI escrito en otro lenguaje de programación como Perl o C. En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP introducido en el mismo, que producirá cierta salida. El código PHP se incluye entre etiquetas especiales de comienzo y final que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviésemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP[5].

3.4.3 Qué se puede hacer con PHP?

Al nivel más básico, 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 mandar y recibir cookies.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP.

Las siguientes bases de datos están soportadas actualmente:

- Adabas D
- Ingres
- Oracle
- dBase
- InterBase
- PostgreSQL
- Empress
- FrontBase

[5] Para mayor referencia consultar la dirección http://www.php.net

- Solid
- FilePro
- mSQL
- Sybase
- IBM DB2
- MySQL
- Velocis
- Informix
- ODBC
- Unix dbm

PHP también soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos.

3.5. SERVIDORES WEB

3.5.1 Apache Web server

Uno de los servicios mas demandas actualmente es el de Internet especialmente el de poder despachar paginas web, este servicio es conocido como servidor http o servidor web. Este debe proveer además de poder mandar la información, poder generar páginas dinámicas para que el servidor que puedan arrojar datos procesados que fueron el resultado de datos introducidos por el usuario[6].

El servidor WEB se encarga de publicar la información hacia el Internet o hacia una intranet.

[6] Para mayor referencia consultar la dirección http://www.apache.org y http://www.apache-ssl.org

El http es el medio de acceso usado por los navegadores para proveer el acceso a un sitio web, el trabajo del servidor consiste en mandar el contenido en respuesta a las peticiones de un URL, el contenido de un sitio web puede ser estático o generado dinámicamente, aunque las pagina estáticas son creadas por el HTML, las interfaces comunes de programación o más comúnmente llamadas CGIs proveen contenidos dinámicos en el servidor web.

Este tipo de servicios ha crecido de manera muy importante desde el surgimiento de las Intranets. Estas son redes locales que permiten accesar información por la misma metodología que en Internet, solo que debe de estar protegida para accesos fuera de esa red.

GNU/Linux ofrece soporte para varios servidores Web dentro de los cuales él más famoso y robusto se llama Apache. Este servidor WEB viene incluido en la mayoría de las distribuciones gratuitamente y con una interfase fácil de instalación.

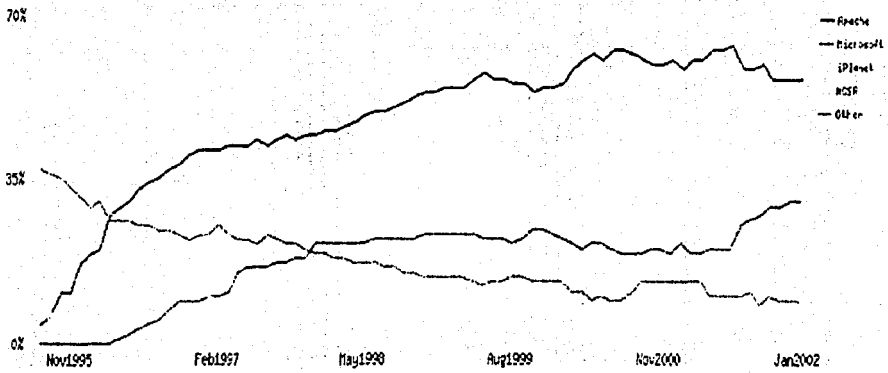
Este servidor http o servidor web es actualmente más usado. Por sondeo y estadísticas se concluyo que aproximadamente el 53% de los servidores web en el mundo son Apache, estas cifras son impresionantes ya que empresas como Netscape que ofrecen su Netscape Enterprise Server o el Internet Information Server de Microsoft no tienen una penetración así, el proyecto Apache permite la utilización módulos para aumentar su funcionalidad sin tener que recompilar el producto.

3.5.2 Estadísticas de servidores Web

El Netcfaft (Web Server Survey), es un software que inspecciona Servidores Web, a Enero del 2002 verificando 36,689,008 sitios obtuvo[7]:

[7] Para verificar estas estadísticas consultar la página http: www.netcraft.com

a) Servidores Web más utilizados de agosto de 1995 a Enero del 2002



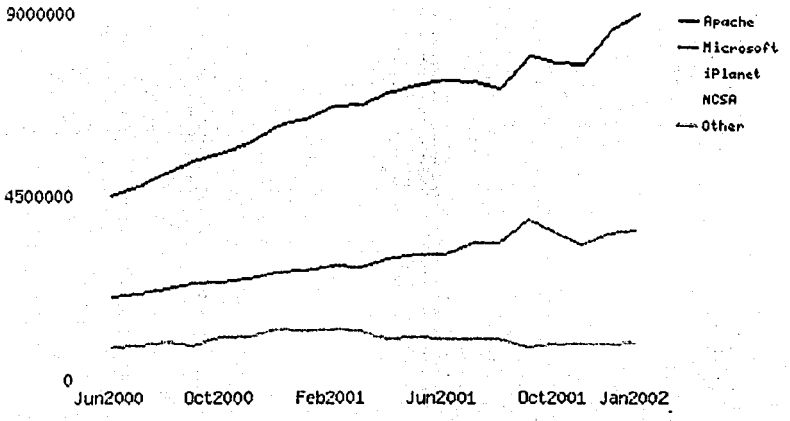
b) Servidores que se utilizan más para el desarrollo Web

Servidor	*Diciembre 2001*	*Porcentaje*	*Enero 2002*	*Porcentaje*	*Cambio*
Apache	20,497,607	56.50	20,866,868	56.87	0.37
Microsoft	11,156,732	30.75	11,097,667	30.25	-0.50
IPlanet	1,302,788	3.59	1,318,991	3.60	0.01
Zeus	792,530	2.18	792,802	2.16	-0.02

c) Sitios Activos

Servidor	*Diciembre 2001*	*Porcentaje*	*Enero 2002*	*Porcentaje*	*Cambio*
Apache	8,588,323	63.34	8,997,645	63.69	0.35
Microsoft	3,609,428	26.62	3,683,141	26.07	-0.55
IPlanet	383,078	2.83	422,806	2.99	0.16
Zeus	172,352	1.27	171,197	1.21	-0.06

d) Total de Servidores Activos a través de todos los dominios de Junio del 2000 a Enero del 2002.



CAPITULO 4 INTERNET

4.1 INTRODUCCIÓN

Internet se puede definir como la más grande colección de redes interconectadas en el mundo. Al usarse genéricamente, internet significa una colección de redes unidas. Un gran ejemplo del potencial de internet ha sido la aparición del World Wide Web.

El World Wide Web(www), es un conjunto de servicios basados en hipermedios, ofrecidos en todo el mundo a través de Internet. No existe un centro que administre esta red de información, sino más bien está constituida por muchos servicios distintos que se conectan entre sí a través de referencias en los distintos documentos, por ejemplo, un documento contenido en un computador en Canadá, puede tener referencias a otro documento en Japón, o a un archivo en Inglaterra, o a una imagen en Suecia.

Al hablar de hipermedios nos referimos a información que puede presentarse utilizando distintos medios, como documentación ejecutable, de texto, gráficos, audio, video, animación o imagen. El WWW fue desarrollado inicialmente en el CERN (el Laboratorio Europeo de Física de Partículas) pero por su extrema flexibilidad ha cambiado mucho últimamente. Cuando una persona ingresa al WWW lo hace mediante un programa "examinador" en general llamado Browser, y a partir de ése momento él esta en el Web.

La parte más importante del trabajo del Web es muy simple. Si usted está esperando encontrar una tecnología compleja y con mucha capacidad, se decepcionará, ya que no lo es. De hecho como en un documento Web preparado por CERN nos indica, la transacción toma lugar en cuatro fases básicas, todas son partes del HTTP (Protocolo de Transferencia de HiperTexto):

1.Conexión

2.Solicitud

3.Respuesta

4.Cierre

En la fase de conexión, el cliente Web (por ejemplo Mosaic, Lynx, Netscape, Explorer) intenta conectarse con el servidor, y entonces aparece la linea de status en el browser e indica la conexión con el servidor http. Usualmente y de hecho cuando la conexión se corta, despliega un mensaje indicando que no se ha podido establecer conexión. Una vez establecido la conexión al servidor de http, el cliente manda una solicitud al servidor. La solicitud especifica que protocolo se ha de usar, además indica al servidor el objetivo que se está buscando y como quiere que el servidor responda. El protocolo puede ser HTTP, pero también puede ser FTP, NNTP (Network News Transfer Protocol), Gopher o WAIS (el protocolo Z39.50).Incluyendo en la solicitud el método, el cual es esencialmente un comando del cliente para el servidor.

Asumiendo que el servidor puede cumplir la solicitud (éste manda mensajes de error cuando no se puede), entonces ejecuta la respuesta. Entonces se ve esta fase de transacción en la linea de status del browser. Como la solicitud, la respuesta indica el protocolo que se está utilizando. Dependiendo del browser, se puede ver exactamente que está sucediendo, representado por el mensaje "Transfering".

Finalmente la conexión se cierra. En esta parte el browser vuelve a accionar nuevamente. Efectivamente, encuentra y despliega los datos solicitados, guarda los datos en un archivo o los despliega en una ventana. Si el objeto es un archivo de texto, el browser desplegará éste como un documento ASCII. Para archivos de imágenes, sonidos y video, el browser los desplegará automáticamente en la pantalla. El browser despliega documentos HTML. Estos

documentos muestran gráficos, sonidos, iconos.

Existen algunas estadísticas que pueden mostrar que potencial tiene Internet[1].

a) El tamaño del Internet

Alexa, un sistema gratuito de ayuda para el navegador determinó que 1.5 millones de Paginas Web nacen cada día. Esto se determinó en función a los dos años que este sistema lleva indexando más de 7 millones de megabytes de Paginas Web.

Otros datos determinados por Alexa son:

- El tamaño aproximado del Internet es de 7 terabytes.
- El Web duplica su tamaño cada 8 meses.
- Existen, aproximadamente 70 millones de Paginas Web
- El 90 % de todo el trafico del Internet circula por 350,000 servidores distintos.
- El 50% de todo el trafico de Internet circula solo en 900 Paginas Web, las más importantes actualmente disponibles.

b) ¿Cuantas personas utilizan Internet hoy en día?

Existen a la fecha (Enero de 2002) 200 millones de usuarios de Internet en todo el mundo de los cuales 114 millones utilizan el servicio WWW (World Wide Web), cifra que va aumentando cada día. La taza anual de crecimiento de usuarios es del 65%.

He aquí un resumen de distribución de usuarios de la Red:

Todo el Mundo	513.41 millones
Africa	4.15 millones
Asia y Pacifico	143.99 millones
Europa	154.63 millones

[1] Para verificar estas estadísticas, consultar la página http://www.khainata.com/extrainternet

Todo el Mundo	513.41 millones
Medio Este	4.65 millones
Canada y EE.UU.	180.68 millones
America Latina	25.33 millones

c) Existen 72 millones de sitios web

Durante los últimos cinco años el número de sitios en Internet ha crecido a una tasa promedio del 61% anual, calculándose que en estos momentos existen más de 72 millones de sitios web en la Red.

Otros datos muentran:

Resolución	
800 x 600	55,00%
1024x768	23,00%
640x480	13,00%

Colores	
16 Bits	53,00%
32 Bits	22,00%
24 Bits (16M)	12,00%
8 Bits (256)	10,00%

d) Los Browsers más utilizados

MSIE 5.x	59,00%
MSIE 4.x	27,00%
Netscape 4.x	13,00%
Tipo Amiga	0,00%
Otros	1,00%
Deconocidos	0,00%

El 96% ya utiliza navegadores de cuarta generación[2]

4.2. APLICACIONES WEB

4.2.1 La pirámide del diseño Web

Las cuatro caracteristicas principales del diseño Web son el contenido, la tecnologia, los aspectos visuales y los económicos. El objetivo principal del contenido es informar, o quizá persuadir a los usuarios. La tecnologia se utiliza para dotar de funcionalidad al sitio. Los temas visuales proporcionan el aspecto del sitio. Finalmente, es necesario considerar, para la mayoria de los sitios, las implicaciones económicas de la construcción del mismo. Si no existe una clara finalidad o beneficio, generalmente el sitio no se construye. La influencia de

[2] Para ver otras estadísticas, se recomendan las siguientes páginas:
http://www.webreference.com/statistics.html
http://www.internetnews.com
http://www.forrester.com
http://www.activmedia.com
http://www.simbanet.com
http://www8.zdnet.com/zdimag/cyberstats

cada característica particular de los sitios Web varía según el tipo de sitio que se construye. Una página principal personal no tiene, generalmente, las mismas consideraciones económicas que un sitio de venta público. Una intranet para una empresa de producción puede que no necesite las mismas consideraciones visuales que un sitio Web público para la promoción de una película, etc. El análisis del diseño Web como un todo unificado es una difícil tarea debido a la necesaria mezcla de disciplinas que dependen de la naturaleza del sitio que se va a construir[3].

Es útil imaginar los sitios Web como pirámides. El contenido proporciona los ladrillos que utilizamos para construir la pirámide, pero los cimientos se apoyan solidariamente tanto en los aspectos visuales como en los tecnológicos, con una fuerte dependencia en los aspectos financieros para que nuestro proyecto merezca la pena. Las arenas movedizas de la tecnología Web convierten en un desafío la construcción de nuestro sitio, es necesario trabajo en equipo y un perfecto conocimiento del medio Web. Incluso siendo suficientemente expertos como para construir un sitio Web atractivo y funcional, nuestros usuarios pueden mirar nuestra atractiva construcción con perplejidad y desconcierto. Los diseñadores, o los que les emplean, utilizan a menudo más tiempo reflexionando sobre sus propias necesidades y deseos que sobre las necesidades y deseos de los visitantes del sitio.

4.2.2 Construcción de sitios Web

La construcción de un sitio Web puede ser muy difícil. Aunque algunas de las tecnologías fundamentales, como HTML, son bastantes fáciles de ominar, los diseñadores parecen cometer abundantes errores. Las principales razones de esta situación son la falta de

[3] POWELL THOMAS.: Diseño de sitios Web Manual de referencia. España, 2001. Mcgraw-Hill. pag 4

experiencia del diseñador, una deficiente definición del proceso y una agendas poco realistas. A causa de las restricciones de tiempo o por inexperiencia, se suele comenzar por un extremo u otro, para después saltar directamente a la implantación sin tener en cuenta los pasos precedentes. Por ejemplo, algunos profesionales comienzan la construcción del sitio a partir de los aspectos visuales. El proceso de diseño consiste en crear modelos de las páginas del sitio mediante una herramienta de diseño visual y después resolver que hacer a partir de eso. Para un sitio catálogo, ésta puede ser la intención, pero, ¿qué sucede con los otros tipos de sitios? Los sitios en los que los aspectos visuales se han desarrollado primero son, por regla general, difíciles de actualizar, carecen de cualquier tecnología importante aparte de los deslumbrantes efectos, su funcionalidad no queda clara y su descarga suele ser lenta. Muchas personas denominan a estos sitios como folletoware o panfletoware, puesto que la función de estos sitios va poco más allá de la de un folleto, la de presentación de un software divertido o la de una introducción en video.

Existen sitios elaborados con un enfoque principalmente tecnológico. Aunque estos sitios pueden utilizar los mejores y más modernos medios de programación, a menudo no tienen un diseño consistente. Algunas veces, estos sitios practican el diseño en árbol de Navidad. Una vez terminada la función, el diseñador decide decorar la página como un árbol de Navidad, con un surtido de imágenes visuales y GIF animados, insertados en la página en un infructuoso intento de hacerla más atractiva. En este caso, los diseños visuales quedan relegados por la tecnología. Al igual que sucede en aquellos sitios que han sido diseñados teniendo principalmente en cuenta el aspecto visual, los sitios de marcado carácter tecnológico se utilizan más para alardear de lo que un diseñador es capaz de hacer que para proporcionar lo que el usuario desea. El uso intensivo de pequeñas aplicaciones y programas complementarios en Java, y el requisito de que los navegadores soporten la última versión

de DHTML, XML o CSS, son signos claros de que las páginas se han diseñado de esa manera, partiendo de, y teniendo principalmente en cuenta, su faceta tecnológica. Este tipo de sitios tienden a practicar un diseño excluyente, utilizan las últimas innovaciones de los navegadores y, frecuentemente, carecen de soporte para la auténtica función del sitio.

4.2.3 Modelo del proceso de desarrollo Web

En primer lugar, hay que considerar la finalidad del sitio. Antes de construir un sitio, se debe definir cuidadosamente el asunto del que se va a ocupar o el objetivo que se está tratando de conseguir. Conocer bien la finalidad reducirá el riesgo de que el proyecto falle y ayudará a determinar si es razonable, incluso, la construcción del propio sitio. Después de determinar el objetivo o los objetivos del sitio, deberá elaborarse una especificación. La especificación incluirá todos los requisitos del sitio y estimará la audiencia de forma muy cuidadosa. A continuación debería realizar un diseño del sitio. El diseño puede incluir tanto el prototipo técnico como el visual. Una vez finalizado el diseño, el sitio debería implantarse y probarse. Finalmente, deberían integrarse y probarse todas las piezas del sitio, incluyendo el contenido. Sólo entonces debería poner el sitio en marcha para, una vez que los usuarios comiencen a enviar sugerencias, realizar arreglos y las modificaciones que sean necesarias. Muchas disciplinas, tales como la ingeniería de software, han definido un modelo de proceso similar al descrito aquí, siendo el más famoso el modelo en cascada, que describe el ciclo de vida del software desde la planificación del proyecto hasta la entrada en servicio final y el mantenimiento. El proceso se divide en varios pasos que guían al diseñador desde los requisitos generales hasta la implantación especifica.

El modelo en cascada básico no es el único modelo en uso, también podrían utilizarse otros modelos de proceso, tales como el modelo en cascada modificado, el espiral, el diseño de

aplicación conjunto (Joint Application Design, o JAD) y otros. Recuerde siempre que el propósito final de un modelo de proceso es servir de guía en el desarrollo del sitio. El modelo ideal deberá ser sencillo y fácil de seguir, ayudará al diseñador a resolver los problemas relacionados con la complejidad del sitio, minimizar los riesgos de fallo del proyecto y proporcionará información para la gestión del mismo.

4.2.4 Utilidad y facilidad de empleo

Los buenos sitios Web son aquellos realmente provechosos para sus usuarios. Puede pensarse en este concepto como en una combinación de utilidad y facilidad de empleo. La utilidad comprende todas las funciones del sitio que satisfacen las necesidades del usuario. La facilidad de empleo es la facilidad de la que dispone el usuario para manejar las funciones del sitio con el fin de conseguir un determinado objetivo. Piense en un sitio cuyo objetivo es proporcionar servicios de banca interactiva, podría decirse que tiene una gran utilidad si proporciona todas las funciones que desean los usuarios, tales como comprobación del saldo, transferencias, pagos de recibos, etc. Sin embargo, el sitio puede ser un terrible fracaso debido a que las funciones que proporciona son tan confusas que no es realmente utilizable. Obviamente, si un sitio no funciona adecuadamente, por errores de diseño o por opciones de diseño desafortunadas que dificultan su uso, no es un buen sitio. Los sitios funcionales serán eficientes y fáciles de manejar y ayudarán a los usuarios a conseguir sus objetivos de manera satisfactoria y sin errores.

4.2.5 ¿Qué es un buen diseño Web?

Sabemos que el desarrollo de sitios Web necesita de mucho talento y de un equilibrio entre la forma, la función, el contenido y la finalidad, es difícil definir exactamente que se entiende por un buen diseño Web. Si se piensa que un proyecto Web tiene cuatro aspectos que están

a menudos enfrentados entre sí, como se presento en la pirámide, no es difícil imaginar que lo que se considera un diseño adecuado variará de un proyecto a otro, incluso de una persona a otra. Los elementos de diseño a tener en cuenta para desarrollar un sitio comercial construido para apoyar un nombre de marca muy conocido son muy diferentes de los de una Intranet. A pesar de la obviedad de la información anterior, muchos diseñadores recurren a guías que afirman contener los conocimientos necesarios para construir cualquier tipo de sitio, con independencias de su objetivo o estilo. A continuación aplican estas ideas ciegamente en sus sitios. Aunque muchas de las ideas son válidas para la mayoría de las sitios, obviamente, no todos los sitios tendrán los mismos objetivos ni intereses. Lo que resulta adecuado para un sitio puede ser un desastre total para otro.

Se considera al diseño Web como un conjunto de reglas indiscutibles que hay que seguir, afirmaciones del tipo no utilice nunca marcos o mantenga siempre el tamaño de la página en 50KB o menos, parecen bastante inofensivos, pero sin un cuidadoso entendimiento de su motivación, los diseñadores están abocados a utilizarlas mal. Considere, por ejemplo, la práctica habitual de mantener el tamaño máximo de cada página Web por debajo de los 50 KB. Este valor se sugiere debido a la velocidad de acceso a la Web. Los usuarios se quejan continuamente de los largos tiempos que necesitan para realizar descargas en los sitios Web. Aunque esto es verdad en el caso de los usuarios que utilizan modems y que acceden a sitios Web muy cargados de gráficos, ¿tienen este problema aquellas personas que dentro de una empresa ven páginas en una intranet, o para aquellas otras que disponen de un sistema de acceso de alta velocidad? Ciertamente, no. Sin embargo, a pesar de esta observación, es frecuente encontrar directrices de diseño, tales como reducir el número de colores utilizados en la imagen, dividir las imágenes en muchas partes pequeñas, etc., en sitios diseñados para entornos de alta velocidad. Las reglas pueden ser buenas, pero es

mejor comprender su motivación y aplicación. Siempre hay que esforzarse en comprender los auténticos motivos que subyacen detrás de las reglas de diseño en lugar de aplicarlas ciegamente.

4.2.6 El proceso de diseño Web

La creación de un buen sitio Web puede ser un verdadero desafío. Con tantos componentes distintos, que van desde el diseño visual a la integración de bases de datos, existe una elevada probabilidad de que algo vaya mal. Para minimizar el riesgo de fracaso de un proyecto Web, necesitamos un método que nos sirva de guía. Desgraciadamente, muchos diseñadores Web parecen utilizar lo que se denomina método NIKE de desarrollo Web, simplemente lo hacen, a menudo con poca previsión y planificación. Esta no es una forma metódica de construir un sitio. Los objetivos del sitio suelen estar definidos de forma poco precisa, el proceso es más intuitivo que planificado y el resultado suele ser impredecible. Los sitios desarrollados de esta manera son como plantas. Crecen de una manera orgánica ocasionalmente producen una hermosa flor, pero lo más frecuente es que se conviertan en una confusa maraña. Los complejos sitios Web requieren una cuidadosa planificación. Siempre debería utilizarse un procedimiento o una metodología que guíase nuestro diseño Web y nuestros trabajos de desarrollo.

4.2.6.1 Método Web ad hoc

La forma en que se construye un sitio Web es sencilla, se realiza el sitio, se efectúa una breve inspección visual con un explorador y, después, se lanza al mundo. Esta forma de proceder es similar al método de codificación y prueba utilizado en pequeños proyectos de software.

En la industria del software, la mayor parte de los profesionales suelen estar de acuerdo en que ese método informal que se diseña según se va avanzado, es adecuado sólo para proyectos pequeños, generalmente con un único programador, y cuando el mantenimiento necesario en el futuro se supone de poca entidad. Con frecuencia, los programas realizados con una planificación tan escasa presentan una lógica de programación enrevesada (a la que se suele denominar código espagueti), cuyo mantenimiento es muy difícil porque nadie, excepto el diseñador inicial, puede desenredar el lío. Los sitios Web pequeños, con una corta esperanza de vida, suelen ser desarrollados por una sola persona utilizando poca metodología. Al inspeccionar el sitio resulta fácil comprobar la existencia de código espagueti como base de la codificación HTML y JavaScript, y de la estructura de la navegación, aderezado todo ello con un plato de ensalada HTML fuertemente removida.

Una buena planificación ayuda a solucionar algunos de los problemas que pueden encontrarse durante el desarrollo de un proyecto Web. Desgraciadamente, en el método ad hoc la planificación se limita, con frecuencia, a unas pocas reuniones, una escasa e incompleta recopilación del posible contenido y, puede que, a un diagrama de flujo realizado apresuradamente. En este caso, el tiempo utilizado en la planificación, generalmente, es despreciable si se compara con el tiempo empleado en el desarrollo. Por supuesto que siempre es posible planificar demasiado y padecer una especie de parálisis de análisis, que mantiene un sitio en construcción permanente. Pero esta situación es relativamente rara. Tenga en cuenta siempre que el tiempo empleado en la planificación debe ser proporcional a la complejidad del proyecto. La clave para superar los desafíos propios de la dirección de un proyecto es crear un procedimiento formal para planificar, probar y desplegar un sitio de una manera estructurada.

4.2.6.2 Modelo del método Web básico

Para disminuir la dificultad de la construcción de sitios Web, deberíamos adoptar un método modelo que describa las diferentes fases presentes en el desarrollo de sitios Web. A continuación, el diseñador puede llevar a la práctica cada uno de los pasos, utilizando las guías y la documentación que le muestre al diseñador el modo de hacer las cosas para asegurarse que cada paso se realiza adecuadamente. Un modelo de procedimiento ideal para el desarrollo Web ayudaría al diseñador a superar la complejidad del sitio, a minimizar el riesgo de fallo del proyecto y a entregar el sitio con rapidez proporcionando la información necesaria para poder realizar toda la gestión durante el proceso. Por supuesto que el método ideal debería ser fácil de aprender y de llevar a la práctica. Este deseo es poco realizar y resulta poco probable que un único método se ajuste adecuadamente a los requisitos específicos de cualquier proyecto.

El método básico utilizado en el desarrollo de sitios Web debería resultar familiar a la mayoría de las personas, el menos en espíritu, puesto que es deductivo. El modelo básico se inicia con una idea del conjunto y se va estrechando hasta llegar a los pasos específicos necesarios para terminar el sitio. En ingeniería de software, a este modelo se le denomina modelo en cascada o, algunas veces, modelo del ciclo de vida del software, porque describe las fases de la vida del software. Este modelo comienza con una etapa de planificación, luego sigue una fase de diseño, a continuación la realización y las pruebas, y termina con la fase de mantenimiento. Estas fases pueden parecer pasos independientes, pero el paso de una etapa a la otra no siempre es evidente. Además, el avance no siempre es hacia la finalización, y, en ocasiones, surge la necesidad de volver a visitar algunos pasos previos cuando en el proyecto se producen cambios no previstos.

Lo bueno del modelo en cascada puro es que obliga a los diseñadores a planificar cada paso

desde el principio, pero ésta es también su mayor debilidad. En un proyecto Web existen, con frecuencia, muchas incertidumbres sobre lo que realmente se necesita para realizar un proyecto, particularmente si el diseñador no tiene mucha experiencia en este tipo de desarrollos.

Otro problema con este modelo es que se supone que cada paso es independiente de los demás. Pero la realidad es que en los desarrollos Web, como en los de software, los pasos tienden a solaparse, a influir en pasos anteriores y posteriores y, con frecuencia, tienen que repetirse. Desgraciadamente, el modelo en cascada puede ser bastante rígido y obligar al diseñador a detener el proyecto si se producen demasiados cambios y a rehacer muchos pasos. En resumen, este método no resulta muy adecuado si hay que realizar muchos cambios. Sin embargo, el modelo en cascada para el diseño de sitios Web sigue siendo muy popular porque es a la vez fácil de entender y de aplicar. Además, la existencia de pasos independientes en el proceso resulta muy atractiva para la dirección, puesto que pueden ser controlados fácilmente y servir de referencia.

4.2.6.3 Modelo en cascada modificado

Un aspecto muy importante del modelo en cascada es que obliga a planificar desde el principio. Sin embargo, debido a la cantidad de pasos necesarios en el procedimiento, muchos diseñadores se precipitan en la ejecución de las primeras etapas y terminan repitiéndolas posteriormente o construyendo un sitio basándose en ideas poco apropiadas. El procedimiento es tan rígido que no soporta mucha exploración y tiene asociados riesgos innecesarios. Una posible mejora es emplear más tiempo en las primeras etapas de la cascada e iterarlas unas cuantas veces, analizando los objetivos y los requisitos del sitio antes de comenzar con las fases de diseño y realización. Debido a la naturaleza cíclica de

este procedimiento, se ha creado un proceso en cascada modificado con remolino, en referencia a los pequeños remolinos que se encuentran, con frecuencia, en la naturaleza antes de llegar a una cascada. Cuando se comienza a trabajar en un proyecto con un alto grado de incertidumbre, será una buena idea utilizar el modelo en cascada.

4.2.6.4 Desarrollo de aplicación conjunta

El último modelo de desarrollo de software que sirve para el desarrollo de sitios Web se llama diseño de aplicación conjunta, o JAD (Joint Application Design). También se denomina prototipo evolutivo porque se basa en la evolución de un prototipo del sitio Web siguiendo una serie de pasos hasta llegar a su forma final. En lugar de crear un prototipo del sitio para comprobar una teoría, se construye un prototipo y se pone a disposición del usuario. El usuario lo comenta y estos comentarios se utilizan como guía para la versión siguiente del prototipo, y se continúa de esta manera hasta llegar al desarrollo final.

El procedimiento se desarrolla en pequeños incrementos comparado con el modelo en cascada. Sin embarga, el modelo JAD tiene algunos serios inconvenientes. En primer lugar, el hecho de que los usuarios vean un sitio sin acabar podría daña la relación entre los usuarios y el diseñador. Incluso, cuando los usuarios desean participar activamente en el proyecto, no debemos olvidar que no son diseñadores.

Los proyectos realizados en el sitio JAD son difíciles de presupuestar, puesto que el número de revisiones es impredecible. Si los usuarios son caprichosos, los costes pueden descontrolarse rápidamente. El concepto básico del JAD es la construcción de un sitio imperfecto numerosas veces, hasta conseguir finalmente el sitio correcto. A pesar de sus inconvenientes, el JAD tiene reservado un lugar en el desarrollo Web, principalmente en los proyectos de mantenimiento. Sin embargo, para los proyectos de desarrollo iniciales, el JAD

debería dejarse en manos de diseñadores experimentados, especialmente de aquellos capaces de conseguir una buena comunicación con los usuarios.

Se han analizado unos cuantos métodos de desarrollo candidatos para guiar el diseño de los proyectos Web. Existen otros muchos que podría servir al diseñador. La construcción de un sitio se inicia con la clara identificación del problema a resolver o con la identificación del objetivo que se trata de conseguir para posteriormente, llegar al resultado de forma clara y consistente. El desarrollo de los sitios debería considerarse de una manera crítica y deliberada más que de forma casual o pasiva. Aunque una aproximación crítica no descarta necesariamente al azar o la inspiración repentina, si nos da la oportunidad de dirigirlos. Los diseñadores no deberían considerar el uso de conceptos de ingeniería en el desarrollo de sitios Web como factores limitativos, sino como algo que puede servirles de guía durante el diseño.

4.2.7 Tipos comunes de sitios Web

Existen tres categorías generales de sitios Web: sitios públicos, sitios extranet y sitios intranet.

Un sitio Web público, un sitio Web de Internet, un sitio Web externo o, sencillamente, un sitio Web, es un sitio cuyo acceso no está restringido explícitamente a ningún tipo de usuarios. Un sitio Web es, en cierto sentido, un lugar público, abierto a todo el que quiera visitarlo en Internet. No todo el mundo querrá visitar el sitio, pero no existe ninguna limitación sobre quién puede visitarle. En el otro extremo del espectro se encuentra el sitio Web intranet, generalmente llamado sólo intranet. Un sitio intranet es generalmente, privado, y con frecuencia, sólo está disponible para los usuarios de una determinada red privada.

Un sitio Web intranet es un sitios que está reservado a una determinada organización y

generalmente funciona dentro de una red privada, en vez de hacerlo directamente en Internet.

En una posición intermedia entre un sitio Web externo y una Intranet se encuentran los sitios semiprivados, denominados generalmente extranet.

Un sitio extranet es un sitio Web al que puede acceder una clase limitada de usuarios, pero a la que se accede a través de Internet.

Un ejemplo de una extranet podría ser un sitio de abastecimiento a los socios de una empresa o a los distribuidores. La limitación del acceso a un sitio extranet puede variar desde las sencilla seguridad por oscuridad, que consiste en no utilizar una dirección o un número de puerto de servidor Web notorios, hasta la restricción de páginas al explorador visitante mediante una clave o una dirección IP.

La diferencia principal entre estas tres diferentes categorías de sitios es la audiencia. Los sitios Web públicos están completamente abiertos, mientras que los extranet e Intranet son más exclusivos. Cuanto más privado sea el sitio mayor conocimiento deberá tener su diseñador sobre los potenciales usuarios. Para una Intranet privada, el diseñador puede reunirse físicamente con los potenciales usuarios del sitio y conocer las capacidades de cada uno de ellos, desde su sofisticación como usuarios de PC, hasta el equipo o el explorador que utilizan.

En el otro extremo del espectro se encuentra el sitio Web público. Los diseñadores de sitios públicos conocen muy poco de sus usuarios, raramente consiguen contactar directamente con ellos y, con frecuencia, tienen poca información sobre sus capacidades. Las consideraciones de diseño variarán enormemente entre los distintos sitios Web, como se refleja en la tabla siguiente:

	Intranets	Extranets	Sitios Públicos
Información sobre los usuarios	Alta	Media	Baja
Capacidad de planificación	Posible	Generalmente posible	Difícil o imposible
Ancho de banda	Alta	Varía	Varía mucho
Capacidad para incluir tecnología	Sí	Algunas veces	Raramente

4.2.7.1 Sitios interactivos frente a sitios estáticos

Los sitios también pueden clasificarse en interactivos y estáticas.

Un sitio interactivo es aquel en el que los usuarios pueden actuar directamente sobre el contenido del sitio o con otros usuarios del mismo.

Hasta cierto punto, todos presentan algún tipo de interactividad, en el sentido de que los usuarios pueden elegir cómo explorar su contenido. Los sitios verdaderamente interactivos permiten a los usuarios manipular el contenido. Un sitio que permite a un usuario incluir cuestiones de soporte técnico para otros usuarios se considera que es interactivo, mientras que un sitio en el que sólo se permite a los usuarios consultar respuestas preexistentes se considera que es estático.

Un sitio estático es aquel cuyo contenido es relativamente fijo, en el que los usuarios no

pueden modificar ni el aspecto ni el ámbito de los datos que observan. En resumen, el visitante no tiene posibilidad de interactuar con el contenido del sitio, excepto en la elección del orden en que desea ver el contenido.

Acceder a un sitio estático es como leer una revista. El usuario puede elegir entre avanzar o retroceder por sus páginas y leer los artículos en cualquier orden, pero la presentación es relativamente rígida y no existe posibilidad de hacer nada con el contenido, excepto leerle.

4.2.7.2 Sitios dinámicos

Aunque, en un momento dado, el contenido de muchos sitios parece relativamente estático, lo más frecuente es que se mantenga en un estado de cambio gradual. Cuando más frecuentes sean estos cambios, más dinámico será considerado el sitio. El contenido de muchos sitios se está actualizando continuamente. Frecuentemente, se incluye en la página una indicación de la fecha del último cambio, para indicar lo reciente que es su contenido.

Un sitio generado dinámicamente es aquel que sus páginas se generan en el momento de la solicitud por parte del usuario.

La ventaja de un sitio generado dinámicamente es que las páginas se crean en función de las condiciones de exploración o los deseos del usuario. Por ejemplo, un sitio estático tiene una única forma de presentación para todos los usuarios, mientras que un sitio dinámico puede presentar múltiples formas, optimizadas para diferentes exploradores o para distintos anchos de banda.

La ventaja de un sitio generado dinámicamente debería ser obvia, puesto que presenta su contenido de la manera que el usuario probablemente desea. Un sitio que tiene como objetivo un determinado tipo de usuarios y les permite determinar exactamente lo que desean ver se denomina, con frecuencia, página personalizada. Los ejemplos de páginas

personalizadas se están haciendo cada vez más frecuentes, las más familiares son, probablemente, las páginas de inicio personales, tales como (www.yahoo.com), que son corrientes en los mayores sitios Web de consumo.

Un sitio personalizado es aquel cuyo contenido está orientado a un determinado usuario, y éste puede, generalmente, determinar explícitamente el contenido, el aspecto o la tecnología contenidos en esa página.

El inconveniente de los sitios generados dinámicamente es que son bastante más complicados de crear y frecuentemente ocupan el servidor de forma intensiva, porque las páginas tienen que generarse para cada usuario en el momento de realizar la visita. Con frecuencia, los sitios generados dinámicamente utilizan una base de datos para almacenar su contenido. En estos sitios, las páginas se forman a partir del contenido, fusionado con plantillas en el momento de la solicitud, para crear la página que se va a entregar. Por el contrario, un sitio Web estático es relativamente sencillo, porque actúa simplemente como un servidor archivo.

4.2.8 Estructuras del sitio

Hay dos tipos de estructuras en cualquier sitio Web: la lógica y la física. La estructura lógica reseña cómo se relacionan entre sí los documentos y define los vínculos entre documentos. Sin embargo, la ubicación lógica de los documentos dentro del sitio no tiene por qué estar relacionada con la ubicación física real del documento. La estructura física describe dónde se encuentra realmente un documento, mostrando, por ejemplo, el camino al directorio del documento en un servidor Web o su situación en una base de datos.

A los usuarios no les preocupa dónde se genera la información, desde el punto de vista del sistema de archivos, mientras que puedan encontrarla en el sitio. Un usuario no necesita

saber qué tipo de datos contiene un determinado disco, ni cómo está organizado su árbol de archivos. Por ejemplo, un archivo determinado podría estar en un profundo directorio de un sistema de archivos con una dirección, tal como D:\Website\DemoCompany\Assests\Product\RobotButler\Index.htm. Sin embargo, el usuario vería simplemente una dirección URL, tal como http://www.democompany.com/RobotButler/. La ventaja de no dar a conocer los caminos o direcciones reales está clara. Si utiliza caminos abstractos, tendrá la libertad de cambiar la posición de los archivos, siempre que ellos mantengan la dirección URL adecuada, conocida por los usuarios. Afortunadamente, todos los servidores Web modernos soportan las facilidades de realización de mapas para crear caminos virtuales, de manera que no es necesario reproducir su estructura lógica en un sistema físico de archivos.

4.2.9 Modelos de organización lógica de un sitio

Existen cuatro formas principales de organización lógica de los sitios Web: lineal, en rejilla, por jerarquías y en tela de araña(Web). También son corrientes algunas variaciones sobre estas formas, como, por ejemplo, la combinación de todas o de algunas de ellas dentro de un sitio grande. Es importante elegir la organización lógica adecuada para un determinado sitio, para que éste sea funcional.

4.2.9.1 Lineal

La estructura lineal es la más familiar de todas las estructuras, porque la información impresa tradicional tiende a seguir este tipo de organización. Por ejemplo, los libros se escriben de manera que una página sigue a la siguiente de una manera lineal. La presentación de la información de una forma lineal es, con frecuencia, muy útil cuando se analiza un procedimiento paso a paso, pero hay ocasiones en las que puede necesitarse algún tipo de

información suplementaria.

4.2.9.1.1 Lineal Básica

Una organización lineal pura facilita la progresión ordenada a través del cuerpo de la información.

En la Web, esta forma es más adecuada para el tipo presentación por diapositivas, con el fin de proporcionar a los visitantes una visión general de la empresa y de sus productos. Utilizando una organización secuencial controlada, tal como una forma lineal, el diseñador se asegura que el usuario recibe la información en el orden deseado.

El sitio de organización lineal es muy predecible, porque el diseñador sabe exactamente a qué sitio irá el usuario en el siguiente paso. Debido a este conocimiento, es posible hacer una precarga de la información siguiente, mejorándose así la percepción del funcionamiento del sitio.

Puesto que la única opción es moverse hacia delante o hacia atrás, el usuario puede encontrar esta organización muy restrictiva, por este motivo, es importante que conozca en qué punto se encuentra, dentro de una estructura lineal, y qué hay en la página anterior y en la posterior a la que se esta viendo.

4.2.9.1.2 Lineal con alternativas

Aunque una organización lineal es útil para presentar la información en un orden predeterminado, proporciona al usuario pocas posibilidades de relacionarse con la información. Una organización lineal con alternativas simula la interactividad proporcionado dos o más opciones para salir de una página que finalmente, lleva al usuario a otra página dentro de la secuencia.

Los usos de estas formas son muy numerosas. Imagine un sitio Web que solicita del usuario, en cada página, un SI o un NO como respuesta a una pregunta, y pasa al usuario a la página

siguiente en función de la persona. Aunque el usuario pudiera pensar que está en juego una cierta tecnología de punta, en realidad, los dos caminos están establecidos de antemano y el usuario sólo tiene una ilusión de interactividad. A pesar de esa apariencia de poder elegir opciones, la estructura lineal con alternativas conserva el camino lineal a través de un documento. Desgraciadamente, con esta forma, la existencia de varios caminos posibles dificulta la precarga de las páginas.

4.2.9.1.3 Lineal con opciones

Una estructura lineal con opciones es adecuada cuando tiene que conservarse el camino general, pero hay que dar cabida a ligeras variaciones, tales como saltarse determinadas páginas. Este tipo de organizaciones de hipertexto puede ser útil para una encuesta en línea en las que determinados usuarios podrían saltarse algunas cuestiones que no les son aplicables. Dado que la estructura lineal con opciones proporciona un medio de saltar hacia delante dentro de la estructura lineal, este tipo de organización se llama también lineal con saltos hacia delante.

Esta organización simula un sistema inteligente, aunque, con frecuencia, no contiene más que páginas estáticas con una estructura de hipertexto bien pensada.

4.2.9.1.4 Lineal con caminos laterales

Una organización lineal con caminos laterales permite algunos desvíos controlados. Aunque el usuario puede tomar un corto desvío lateral, la estructura le obliga a regresar el camino principal.

Este desvío de la progresión lineal es equivalente a una columna anexa a un artículo de una revista. En lugar de alejar demasiado al usuario del camino principal, esta pequeña información adicional enriquece la experiencia. Si esta nota lateral se incluyera dentro de la progresión lineal principal, la continuidad del mensaje principal quedaría desdibujada. Sin

embargo, cuando se añaden muchos caminos laterales, la estructura comienza a parecerse más a una estructura jerarquizada, o en árbol.

4.2.9.2 Rejilla

Una rejilla es una estructura lineal, en la que sus elementos se relacionan tanto vertical como horizontalmente. Puesto que una rejilla tiene una organización espacial, es ideal para conjuntos de elementos que están relacionados entre sí, sin embargo, una estructura pura en rejilla es, hasta ahora, poco frecuente en la Web. Cuando está bien diseñada, una rejilla ofrece orientación horizontal y vertical para que el usuario no se sienta perdido dentro del sitio.

Aunque una estructura en rejilla es muy regular y su exploración puede resultarle muy sencilla al usuario, no hay muchos tipos de información que sean lo suficientemente regulares como para adaptarse bien a este tipo de organización. Una notable excepción son los catálogos de productos.

4.2.9.3 Jerarquía

La estructura hipertextual más común en la Web es la jerárquica, o en árbol. Aunque esta estructura no puede proporcionar la estructura espacial de una rejilla o la predecibilidad y control de una estructura lineal, la estructura jerárquica es muy importante porque puede modificarse para esconder o mostrar tanta información como se desee. Esta estructura comienza con una página raíz que, frecuentemente, es la página principal del sitio o sección. La página principal o raíz de un sitio con estructura en árbol sirve como página de referencia y, como tal, con frecuencia su aspecto es muy diferente del de las otras páginas del sitio. Las páginas de referencia del sitio son clave para una exploración con éxito. A partir de la página principal se presentan varias opciones. A medida que el usuario se introduce más profundamente en el sitio, las opciones serán más y más específicas, hasta que, finalmente,

se alcanza un destino u hoja del árbol. Por este motivo, estos árboles suelen describirse por su profundidad y anchura.

4.2.9.3.1 Árboles estrechos

Un árbol estrecho ofrece solamente unas pocas opciones, pero pueden ser necesarios muchos clics para alcanzar el destino final, en esta organización predominan la profundidad sobre la anchura. En un árbol estrecho, el usuario tendrá que escoger muchas opciones antes de alcanzar una página hoja, pero, en algunos sitios, éste es un medio muy eficaz de dirigir a los usuarios hacia la información adecuada.

Una forma de saber si un sitio jerarquizado es demasiado estrecho es cuando, además de la página principal, existen muchas otras páginas que son puramente para exploración. Solo hay que recordar que los usuarios buscan un beneficio y hacer clic de forma interminable por páginas y páginas lo único que proporciona es frustración.

4.2.9.3.2 Árboles anchos

Un árbol ancho o jerarquía ancha se basa en una amplia gama de opciones. Su desventaja principal es que pueden presentar demasiadas opciones porque sus páginas contienen numerosas posibilidades de elección.

Aunque el usuario tiene que hacer clic sólo una o dos veces para alcanzar el contenido, el tiempo que emplea eligiendo entre las numerosas opciones iniciales puede ser contraproducente. Mucha gente piensa que todo lo que es importante debe estar en la página principal.

Sin embargo, si todo tiene un vínculo en la página principal, entonces no se conserva la jerarquía y la información puede perder su efectividad.

4.2.9.3.3 Árboles Web

La realidad de la Web es que las estructuras jerárquicas puras típicas se utilizan muy poco.

En un árbol perfecto no existen vínculos cruzados y, para alcanzar otras partes del árbol, es necesario retroceder. Si un usuario se encuentra en la página A de la estructura representada más abajo, para alcanzar la página B tendrá que retroceder dos niveles y después continuar hacia delante.

Aunque es posible retroceder en la Web mediante el botón Atrás del explorador, con frecuencia, se añaden vínculos a las páginas para que los usuarios que llegan a ella, por un camino distinto del principal, puedan explorar el sitio. En muchos casos, las páginas están vinculadas transversalmente mediante una barra de exploración o vínculos explícitos de retroceso, para ayudar a los usuarios a explorar rápidamente la estructura del sitio.

4.2.9.3.4 Malla completa

La estructura de un sitio en el que cada una de sus páginas está vinculada a todas las demás se denomina malla completa.

En una malla completa, el número de vínculos es igual al número de páginas por el número de páginas menos una. Esto quiere decir que en un sitio de cinco páginas habrá 20 vínculos, en un sitio de 10 páginas habrá 90 vínculos, en un sitio de 100 páginas habrá 9,900 vínculos (100 * 99) y en un sitio de 1,000 páginas habrá casi un millón de vínculos. Una malla completa no funciona demasiado bien desde la perspectiva de la facilidad de uso. Si admitimos un máximo de nueve vínculos por página sólo podremos tener un sitio con un máximo de 10 páginas con una estructura de malla completa. En realidad, la mayoría de los sitios tienden a utilizar estructuras en malla parcial, con vínculos cruzados, únicamente entre las páginas más importantes.

4.2.9.3.5 Formas mixtas

Mientras que una estructura en árbol ancho puede presentar demasiada información, una jerarquía estrecha puede esconder también demasiada información. Una estructura lineal

proporciona muy poco control al usuario, mientras que una estructura Web pura le proporciona demasiado. En algunos casos, será necesario aumentar la jerarquía para permitir que las opciones lleguen hasta su parte final. Esta estructura se denomina de forma mixta o de jerarquía mixta, puesto que el árbol es la forma dominante de la estructura. Una forma mixta es, probablemente, la forma de organización más corriente de todas las utilizadas en la Web. Dentro de una forma mixta pueden encontrarse dispositivos lineales, saltos e incluso retículas. Piense en un sitio que dispone de un botón Descargue Ahora o algo similar, que salta hasta lo más profundo de su estructura. Esta forma de proceder se parece algo a una estructura lineal con saltos. Otros sitios pueden tener recorridos lineales, disponibles únicamente desde ciertas páginas del sitio. Aunque su organización espacial no es tan pronunciada como en otras estructuras, la utilización de algún tipo de jerarquía sigue siendo aún evidente en la mayoría de los sitios mixtos.

Un estilo mixto muy corriente consta de una estructura lineal para entrar a un sitio, y de una estructura en árbol a partir de la página principal real. Los sitios que disponen de páginas de bienvenida o de recorridos que llevan a una página central, desde la que el usuario puede explorar, utilizan este tipo de estructura.

Otro sitio, que no es realmente único, se denomina estructura de centro y radios (hub and spoke). Algunos sitios constan de páginas principales, a las que se denominan centros, y de subpáginas, a las que se llega a través de los radios. Para visitar otras páginas del sitio, el usuario siempre se ve obligado a volver a la página central. Muchos portales utilizan esta estructura para fomentar las vistas a estas páginas. Sin embargo, no existe realmente diferencia entre un modelo de centro y radios y un modelo de árbol típico.

La única ventaja de la estructura de centro y radio es que proporciona una forma sencilla de visualizar el sitio. Por ejemplo, algunas herramientas de generación de planos de sitios

generan los mapas de los sitios utilizando este estilo porque son más sencillos de ordenar que una estructura de árbol.

4.2.9.4 Estilo Web

Cuando existen demasiados vínculos cruzados, saltos hacia delante y otros tipos de artimañas en un conjunto estructurado de documentos, puede que su forma no quede clara para el usuario. Cuando un conjunto de documentos parece no tener una estructura perceptible, se le denomina Web pura.

Una estructura de este tipo puede ser difícil de utilizar porque carece de una clara orientación espacial. Aunque se puede acceder rápidamente a la información si se eligen las opciones adecuadas, es difícil orientarse en un sitio Web con una estructura tan poco clara. Si la estructura de un sitio no está clara o no le es familiar al usuario, éste puede recurrir a una explotación con base en la página principal, regresando siempre al nivel superior cada vez que se comienza una nueva tarea. Sin embargo, la ventaja de una forma menos estructurada es que proporciona una gran cantidad de expresividad.

4.2.10 Criterios para la evaluación de Páginas Web

¿Qué es lo que hace que una Página Web tenga éxito? - Todos coinciden en que el éxito radica en la repetición de visitas, es decir, que un visitante vuelva a la página repetidas veces. SurveySite desarrolló una lista de 40 factores que indican el mayor impacto sobre la repetición de visitas a un Home Page. Desde este análisis, una encuesta final con 12 preguntas desarrolló el siguiente detalle[4]:

1.El factor más alto (74%) de probabilidad para lograr una visita repetida a un sitio es el CONTENIDO. La razón principal para no volver a un sitio era "Contenido frivolo".

[4] Para mayor referencia consultar la dirección http://www.laguiainternet.com

2.El segundo factor importante es lo ameno del sitio (71%), proveyendo una diversión o experiencia novedosa, o una interesante interface, etc.

3.Los terceros y cuartos factores importantes eran la calidad de la organización, o esquema (68%).

4.El grado de originalidad (66%). Muchas respuestas indicaban que no volverían al sitio porque era "como todos los demás".

5.Un 64% por la facilidad de encontrar información.

6.Seguido por visitadores que pensaron que su visita fue "excitante" (61%).

7.Visualmente atractivo (58%).

8.Fácil de navegar (54%).

9.Y la velocidad a que ellos podrían moverse alrededor del sitio (53%).

La calidad de gráficas no era una consideración importante cuando se comparó con otros factores. Interesante ver que los visitantes no encuentran la calidad de gráficas muy importante, trauma de la mayoría de los diseñadores de Páginas Web.

4.2.11 El futuro cercano

Predecir el futuro puede ser difícil. Hablar sobre el futuro lejano puede ser una apuesta segura, en parte porque cuando llegue ese futuro será muy difícil verificar las afirmaciones hechas. En el caso del futuro lejano de Internet y de la Web digamos de aquí a 25 años o más, la mayoría de los expertos predicen que los servicios de datos de alta velocidad estarán disponibles en una gran parte del mundo desarrollado, tanto en formas móviles como fijas, y con una gran variedad de interfaces, incluyendo órdenes por voz, escritura y potencialmente, incluso, biomecánicas. Como en una representación de ciencia ficción, es posible que podamos movernos por el mundo real y relacionarnos con el mundo virtual casi al mismo

tiempo. La interacción con la información será posible mediante interfaces de voz que, de forma remota, controlen su casa o su empresa, e, incluso, su mismo entorno, tanto sin conexión como en línea, se adaptará a sus necesidades y hábitos. El camino hacia el futuro es un camino largo. La mayoría de la gente está, probablemente, más interesada con lo que va a suceder en el futuro cercano.

Lo que va a suceder en el futuro inmediato no siempre es fácil de predecir, pero hay unas cuantas cosas que parecen bastante evidentes. Nos centraremos en algunas que son importantes para el diseñador Web antes de hablar sobre otras importantes tendencias que pueden cambiar drásticamente la forma en que se desarrollan los sitios.

CONCLUSIONES

El sistema que se implemento, fue desarrollado como ya hemos mencionado con ciertas herramientas, de las cuales consideramos lo bastante buenas para el desarrollo. Además de haberlo implantado bajo internet, que actualmente es lo más viable ya que la tendencia de todos los desarrollos van enfocados a ese ambiente.

También conforme a la metodología UML que fue utilizada podemos decir, que no es simplemente un lenguaje para modelamiento orientado al objeto, sino un "lenguaje para modelamiento unificado" relativo a sistemas en general. Es un lenguaje para modelamiento de propósito general evolutivo, ampliamente aplicable, posible de ser soportado por herramientas e industrialmente estandarizado. Se aplica a una multitud de diferentes tipos de sistemas, dominios y métodos o procesos.

Además ya están disponibles herramientas para soportar la aplicación del lenguaje, para especificar, visualizar, construir y documentar sistemas. Como un lenguaje para modelamiento industrialmente estandarizado, no es un lenguaje cerrado, propiedad de alguien, sino más bien, un lenguaje abierto y totalmente extensible reconocido por la industria. UML facilitar el incremento del valor, aumentando la calidad, reduciendo costos y reduciendo el tiempo de presentación al mercado; menejando riesgos y siendo proactivo para el posible aumento de complejidad o cambio.

El cambio de una metodología a otra (migración) no pretende ser traumático, es decir, no deberá haber cambios importantes. Esto es debido a que el nuevo lenguaje de especificación no supone un cambio radical con respecto a las otras metodologías existentes. Pero una cosa que no debe olvidarse es que un estándar no se convierte en tal porque sus creadores

lo decidan, sino que conlleva un tiempo y la colaboración de muchas personas interesadas en mejorar su trabajo.

El objetivo central del lenguaje es abstraer cualquier tipo de sistema, sea informático o no, mediante diagramas. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibujan de forma que resalten los detalles necesarios para entender el sistema. La manera en que se modela un sistema para ser construido y la elección de los puntos importantes que se reflejan en el modelo junto con los desechados determinan cómo se solucionará el problema y la dificultad de su posterior implementación.

El objetivo de este modelo es que sea un modelo unificado, abierto, que siga evolucionando en conjunto y no por separado, tal como estaba ocurriendo hasta ahora, comprensible por el hombre, utilizable por la computadora y fundamentalmente con la capacidad de unificar las perspectivas de diferentes sistemas.

Se hace otra vez mención que aunque UML es orientado a objetos preferentemente, es útil en cualquier modelo tecnológico ya que es independiente de lenguajes de programación o tecnología determinada.

El modelado sirve no solamente para los grandes sistemas; aún en aplicaciones de pequeño tamaño se obtienen beneficios de modelar, sin embargo, es un hecho que entre más grande y más complejo es el sistema, el modelado juega un papel más importante. A través del modelado reducimos el ámbito del problema de estudio al enfocar solo un aspecto a la vez.

Las reglas de simbología que aplican a cualquier tipo de modelo hecho bajo este lenguaje, por ejemplo, el modo en que se coloca un comentario en cualquier diagrama o el modo en que se aumenta la nomenclatura existente en UML. Diferentes tipos de diagramas están diseñados para enfocar un aspecto en particular de un sistema. En un modelo de análisis muestra los conceptos de negocios del sistema, sus relaciones, sus datos(atributos) y

operaciones.

UML puede ser usado extensivamente en: Recopilación de requerimientos, Análisis de aplicaciones, Diseño de sistemas, en pruebas, en implementación, en reingeniería y prácticamente en cualquier actividad de desarrollo que sea susceptible de ser modelada.

En cuanto a las herramientas como lo es GNU/Linux es un sistema operativo ideal para aplicaciones de red que requieran un funcionamiento robusto de forma continuada. Este es el caso de los servidores de bases datos, programas que permiten el acceso a los datos de una empresa, de un proyecto o de cualquier tipo de organización, datos normalmente vitales.

La robustez y eficiencia de Linux ha hecho que empresas hayan portado bases de datos del renombre a Linux, y a precios muy inferiores a otros entornos. Así GNU/Linux se afianza en otro nuevo campo del mundo software, un campo muy importante y que puede ayudarle a multiplicar una confianza que comienza a palparse dentro del mundo empresarial.

Al utilizar las herramientas de libre distribución en nuestro caso nos permitio haber realizado el sistema sin ningún costo, logrando además un sitio con un mayor grado de seguridad, utilizando una menor cantidad de recursos en la administración.

Apache, otra de las herramientas utilizadas reconociendo también que es el servidor de web por excelencia. Ha sido uno de los mayores éxitos del software libre y su supremacía entre los servidores de web. Su desarrollo en continuo y su portabilidad le han llevado plataformas como Windows NT, Windows 95 y OS/MAC, a parte de casi todos los sistemas Unix.

Una de las principales características de Apache es su extensibilidad basada en una gran modularidad de su código fuente, facilitando enormemente la programación de aplicaciones en el lado del servidor, especialmente en el campo de acceso a bases de datos.

Al utilizar PHP nos dimos cuenta que presenta múltiples ventajas frente a otros lenguajes de

programación que necesariamente harán que este lenguaje se imponga como una alternativa para el desarrollo de todo tipo de aplicaciones.

En primer lugar se ejecuta a través de una interfaz que le resulta familiar al usuario: el cliente web. Tampoco es necesario tener que instalar ningún software adicional en la estación cliente para usar un programa PHP aparte del propio navegado web.

De la misma forma, la ejecución de un programa PHP se puede realizar desde un cliente web de cualquier plataforma: el usuario puede escoger su sistema operativo y su cliente web preferidos.

El propio diseño de PHP lleva incorporada esta virtud. El programa se ejecuta en un servidor al cual puede acceder desde cualquier puesto de una red. El servidor siempre podría limitar el acceso a sólo determinados puestos y además obligar a la autenticación de un usuario para poder acceder a ciertas partes de un programa.

Cuando configuremos el servidor tenemos que asegurarnos de que se han tomado las medidas de seguridad necesarias para impedir accesos indebidos al sistema.

Hay múltiples razones para escoger PHP como lenguaje de programación para entornos web en comparación con otros existentes en el mercado. Lo más probabe es que cuando tenga cierta experiencia con PHP usted mismo sea quien dé las razones de por qué usar este lenguaje, mientras tanto podemos apuntarles estas:

- Es muy fácil de aprender. Siempre y cuando se tenga los conocimientos básicos de programación.
- Se puede hacer cualquier cosa con PHP. Bueno, no espere que lo haga solo.
- Está ampliamente probado como herramienta. Un millón y medio de servidores por el momento, en todo el mundo lo avalan como una plataforma para desarrollar

aplicaciones de portales, comercio electrónico, aplicaciones en intranets, etc.

- Porque se puede usar prácticamente en cualquier plataforma.
- Porque, aunque no dispone de soporte comercial, existen numerosas listas de correo en las que se pueden obtener soluciones a los problemas que se puedan presentar.
- Porque existen recursos en la web que pueden facilitar el desarrollo de nuestras aplicaciones. Existen bibliotecas de clases que resuelven los problemas más frecuentes con los que se puede encontrar el programador.
- Porque no requiere unos recursos desmesurados para funcionar. Es posible tener PHP funcionando en un portátil 486 con 16M de RAM para desarrollo. En un equipo más grande estará más cómodo, por supuesto.

La facilidad con que PHP es capaz de consultar una base de datos le configure una potencia excepcional a la hora de generar páginas web dinámicas.

No obstante hay que reconocer que la gran mayoría de las aplicaciones desarrolladas con PHP con un gestor de base de datos utiliza Mysql, un sistema gestor eficiente, suficiente para la mayoría de las aplicaciones que se desarrollan. Además las últimas versiones que han surgido parecen muy prometedoras.

El acceso a base de datos es una de las características más importantes de las que dispone PHP. La generación de contenidos dinámicos normalmente se basa en la información contenida en base de datos que se consultan en función de los datos solicitados por el usuario y devueltos a través de las interfaz web.

Quizás una de las características más interesantes que incorpora es la facilidad para consultar bases de datos y generar páginas en función de los resultados obtenidos en la correspondiente consulta. Las consultas se pueden realizar, bien en modo nativo o, bien,

mediante ODBC. En modo nativo, es compatible con Oracle, Postgres, Informix, MYSQL, y muchas otras.

Además, en este lenguaje podemos definir clases y usar ciertas características de la programación orientada a objeto.

Por último queremos señalar que nos dio una gran satisfacción no solamente porque sirvió como proyecto de tesis y con esto la posibilidad de Titulación. Sino también el aprendizaje que se adquirió durante el desarrollo del sistema. Por ambas partes, además de poder aportar algo para la Universidad Nacional Autónoma de México, en especial a la Facultad de Estudios Superiores Cuautitlán Campo 4, que sirva de apoyo a la Licenciatura en Informática. El haber aprovechado los recursos con los que cuenta la Facultad de Estudios Superiores les damos las gracias, y esperamos sea de gran utilidad.

BIBLIOGRAFÍA

1. MEYER BERTRAND.: Object Oriented Software Construction (OOSC), Prentice-Hall, Englewood 1997, 950 pp.

2. BOOCH GRADY.: Análisis y Diseño Orientado a Objetos con Aplicaciones, Addison-Wesley Días de Santos 2ª. Edición, México 1995, 870 pp.

3. JOYANES AGUILAR LUIS.: Programación Orientada a Objetos, McGraw-Hill 2ª.Edición, España 1998, 895 pp.

4. FOWLER MARTÍN, SCOTT KENDALL.: UML gota a gota, Pearson Educación, México 1999, 203 pp.

5. SCHMULLER JOSEPT.: Aprendiendo UML en 24 horas, Prentice-Hall, México 2000, 423 pp.

6. CASTAÑO ADORACIÓN DE MIGUEL.: Fundamentos y Modelos de Bases de Datos, Alfaomega 2ª. Edición, México 1998, 515 pp.

7. KORTH HENRY F., SILBERSCHATZ ABRAHAM.: Fundamentos de Bases de Datos, McGraw-Hill 2ª. Edición, México 1993, 739 pp.

8. MARLAKOWSKI MARK, BUTCHER TONY.: Aprendiendo MySQL en 21 dias, Pearson Educación, México 2001, 533 pp.

9. FABREGA PEDRO PABLO.: PHP 4, Prentice-Hall, España, 349 pp.

10. POWELL THOMAS A.: Diseño de Sitios Web Manual de Referencia, McGraw-Hill, España 2001, 855 pp.

11. RAYA CABRERA JOSE LUIS, MORENO GUTIERREZ JOSE ANTONIO.: HTML 4 Guia de Refrencia y Tutorial, Alfaomega, México 1998, 650 pp.

12. KOLBECK RAINER.: El Gran Libro de JavaScript, Marcombo Data Becker, España 1997, 415 pp.

13. OROS JUAN CARLOS.: Diseño de Páginas Web interactiva en JavaScript, Alfaomega 2ª. Edición, Colombia 2000, 305 pp.

OTRAS REFERENCIAS

1. http://www.rational.com
2. http://www.gnu.org
3. http://www.linux-mandrake.com
4. http://www.linux.com
5. http://www.mysql.com
6. http://www.php.net
7. http://www.apache.org
8. http://www.apache-ssl.org
9. http://www.netcroft.com
10. http://www.khainata.com
11. http://www.webreference.com
12. http://www.internetnews.com
13. http://www.forrester.com
14. http://www.activemedia.com
15. http://www.simbanet.com
16. http://www8.zdnet.com
17. http://www.laguiainternet.com
18. http://www.lawebdelprogramador.com
19. http://www.lavariable.com

GLOSARIO

Aquí se especifican los términos más comúnmente utilizados dentro del proyecto de tesis.

A

Atributo Es cada una de las propiedades o características que tiene un tipo de entidad o de interrelación.

B

Base de Datos Colección o depósito de datos integrados, almacenados en soporte secundario y con redundancia secundaria, apoyándose en un modelo de datos, que han de ser compartidos por diferentes usuarios y aplicaciones

C

Clase Conjunto de datos y métodos.
Conjunto de reglas de creación y comportamiento de los objetos.

Clase abstracta Clase de apoyo, que construimos solo para derivar de ella otras clases, pero de la que no se puede hacer ninguna instanciación.

Clase hija Clase que es derivada directamente de otra. Decimos que la clase

Es hija de la clase A si B deriva directamente de A (está conectada directamente en el árbol de jerarquías de las clases).

Clase padre La clase de la cual otra deriva directamente. Decimos que la clase A es padre de la clase B si B deriva directamente de A (está conectada directamente en el árbol de jerarquías de las clases).

Copyleft Es un concepto general, para proteger actualmente un programa con copyleft, necesita usar un conjunto específico de términos de distribución. Hay muchas maneras posibles de escribir términos copyleft de distribución

D

Distribución Es un agrupamiento del núcleo del sistema operativo Linux (la parte desarrollada por Linus Torvalds) y otra serie de aplicaciones de uso general o no tan general.

E

Encapsulación Vínculo que existe entre los datos de un objeto y los métodos que los utilizan.

Entidad Es una persona, lugar, cosa, concepto o suceso, real o abstracto,

de interés. Es aquel objeto acerca del cual queremos almacenar información en la base de datos.

Extranet Es un sitio Web al que puede acceder una clase limitada de usuarios, pero a la que se accede a través de Internet.

H

Herencia Capacidad que tienen las clases derivadas o hijas para utilizar toda la información (datos y métodos) de todas las clases superiores.

Herencia múltiple Cuando una clase hija deriva de más de una clase padre; con lo que adquiere los datos y métodos de todas las clases de las cuales deriva.

I

Instanciar Genéricamente, crear una nueva variable. En POO, utilizamos este término para referirnos a la acción de crear un nuevo objeto. Este objeto contendrá espacio para almacenar valores para todos los datos de la clase a la que pertenece, y las referencias a los métodos de su clase -funciones miembro-.

La finalidad que se persigue dejando los métodos en la clase, en

lugar de incorporarlos en cada uno de los objetos que instanciemos, es la de ahorrar memoria no duplicando innecesariamente esta información.

Internet

Se puede definir como la más grande colección de redes interconectadas en el mundo. Al usarse genéricamente, internet significa una colección de redes unidas.

intranet

Es un sitios que está reservado a una determinada organización y generalmente funciona dentro de una red privada, en vez de hacerlo directamente en Internet.

M

Método

Se les llama así a las funciones que realizan operaciones con los datos de un objeto. También llamados *funciones miembro*.
Los datos privados de una clase sólo pueden ser modificados por medio de las funciones miembro -todos los métodos de esa clase y sólo los de esa clase-.

Mensaje

El modo de solicitar que un objeto realice una acción, es enviándole un mensaje. Un mensaje normalmente activa cualquiera de los métodos de la clase a la que pertenece el objeto.

Aunque hay mensajes que no activan ningún método, como son los de inspección y/o modificación de un dato del objeto. Para enviar un mensaje a un objeto se utiliza el operador de envío (normalmente '.' o ':', dependiendo del lenguaje).

MySQL

Es un Sistema de Gestión de Base de Datos Relacional, de código abierto esto quiere decir que es accesible para cualquiera, para usarlo o modificarlo.

O

Objeto

Cualquiera de la instanciaciones que hacemos de una clase.
Una clase podemos considerarla como un conjunto de datos. métodos, es decir, como un nuevo tipo de dato (como lo son los tipos carácter, lógico, array, ...).
Un objeto sería una variable cuyo tipo es el la clase a la que pertenece.

POO

Object Oriented Programming System (Programación Orientada a Objetos).

P

PHP

Hypertext Preprocessor, es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Polimorfismo Capacidad que tienen los objetos de clases diferentes para actuar frente al mismo mensaje de formas distintas.

R

Relación Asociación o correspondencia entre entidades.

Reusabilidad Denota la permisividad de un sistema de programación para poder utilizar partes de código ya escritas en problemas diferentes. La POO es el sistema de programación que proporciona un mayor grado de reusabilidad del código.

S

SGBD El Sistema de Gestión de Bases de Datos, es el conjunto de programas que permiten la implementación, acceso y mantenimiento de la base de datos. El SGBD junto con la base de datos y con los usuarios, constituyen el Sistema de Base de Datos.

Sitio dinámico Un sitio generado dinámicamente es aquel que sus páginas se generan en el momento de la solicitud por parte del usuario.

Sitio estático Es aquel cuyo contenido es relativamente fijo, en el que los usuarios no pueden modificar ni el aspecto ni el ámbito de los datos que observan.

Sitio interactivo	Es aquel en el que los usuarios pueden actuar directamente sobre el contenido del sitio o con otros usuarios del mismo.
Sitio personalizado	Es aquel cuyo contenido está orientado a un determinado usuario, y éste puede, generalmente, determinar explícitamente el contenido, el aspecto o la tecnología contenidos en esa página.
Sitio Web público	Es un sitio cuyo acceso no está restringido explícitamente a ningún tipo de usuarios.
Software Libre	Se refiere a la libertad de los usuarios de correr, copiar, distribuir, estudiar, cambiar y mejorar el software.
Sobrecarga	Algunos lenguajes de OOP permiten realizar lo que llamamos sobrecarga de operadores y de funciones. Por sobrecarga de operadores entendemos la posibilidad de definir para un operador existente comportamientos distintos dependiendo del tipo de los datos con los que trabaje el operador. Por sobrecarga de funciones entendemos la posibilidad de definir distintas funciones con el mismo nombre, pero con diferente conjunto de argumentos. Sin sobrecarga de funciones no puede haber *Polimorfismo*.

SQL

Es un método basado en un potente lenguaje, para organizar, administrar y consultar datos almacenados en una computadora. SQL es una sigla que deviene de su nombre en inglés "Structured Query Language" (Lenguaje de Consulta Estructurado).

Superclase

Cualquier clase de la que derivan una o más clases. Normalmente, a la clase que se halla directamente por encima de otra determinada, la llamamos clase padre, y aquella de la que derivan todas -la que se halla a la raíz del árbol de jerarquías- la llamamos *Superclase* o *Clase Base*.

Subclase

Cualquier clase que es derivada de otra (u otras si el sistema permite herencia múltiple) es una subclase.
También llamada Clase Hija o Clase derivada.

V

Variable de clase

Son aquellas de las que solo existe una copia para todas las instancias de la clase. Es decir, todas las instancias de la clase comparten la misma variable en lugar de una copia de la variable que es lo normal.

W

www

El World Wide Web, es un conjunto de servicios basados en hipermedios, ofrecidos en todo el mundo a través de Internet.

ANEXO 1. CASO PRACTICO

El Sistema de Seguimiento a egresados se podrá acceder desde la página de la Licenciatura en Informática con la siguiente dirección: http://informatica.cuautitlan2.unam.mx por medio de la opción "Egresados".

En un inicio se hará una invitación a los egresados de la Licenciatura con los que actualmente se tiene información, y de esta manera se pueda informar a otros egresados con los cuales ellos tengan contacto; y puedan registrar su información.

El coordinador de la Licenciatura en Informática será el responsable del mantenimiento de la información, así como los respaldos y administración del sistema.

La construcción del Sistema de Seguimiento a Egresados, esta conformado por cuatro fases que son las siguientes:

a). ANALISIS

Dentro de esta fase se contempla la información que se recopilo con la asesoría del Lic. Carlos Pineda Muñoz coordinador de la Licenciatura en Informática.

Requerimientos:

- El sistema tendrá dos tipos de usuarios (egresados, administrador).
- Validar el acceso al usuario
- Funciones especiales por cada tipo de usuario.

Funciones para el usuario egresado

- Registro ó Acceso al sistema
- Registro y Actualización a sus datos personales
- Registro y Actualización a sus datos académicos
- Registro y Actualización a sus datos laborales
- Registro y Actualización de publicaciones que haya realizado
- Registro y Actualización de eventos en los que haya participado
- Permitir el envío de comentarios, dudas ó sugerencias
- Consultar avisos que ingrese el administrador al sistema
- Acceso a Reportes
- Acceso al Chat
- Acceso al Forum

Funciones para el usuario administrador

- Mantenimiento de Catálogos que son los siguientes:

  Usuarios, Escuelas, Medios de publicación, Niveles de estudio, Areas, Generaciones.

- Mantenimiento de Avisos
- Acceso a Consultas

Se analizará la información bajo la Metodología UML, la cual fue mencionada en el Capitulo 1

Se hizo uso de varios diagramas como:

Diagrama de Casos de uso: Permite especificar de una manera más clara y precisa cada uno de los requerimientos que debe contener el sistema.

Ver diagrama de Casos de Uso.

Diagrama de Secuencias: Permite especificar la interacción de cada uno de los elementos que contiene el proceso.

De los cuales mencionamos los más utilizados en el proyecto.

Ver diagramas de secuencia

Diagrama de Clases: Describe la vista estática de un sistema en términos de clases y relaciones entre ellos.

Ver diagrama de Clases.

En base al diagrama de clases, se procede a realizar el modelo entidad relación y posteriormente la definición de tablas

Ver el Modelo Entidad Relación y Definición de tablas.

Tenemos conocimiento de que el Diccionario de datos, es un documento que se genera durante está etapa. El cual, es muy importante para la construcción del sistema, ya que nos muestra la estructura de la información que manejará el sistema.

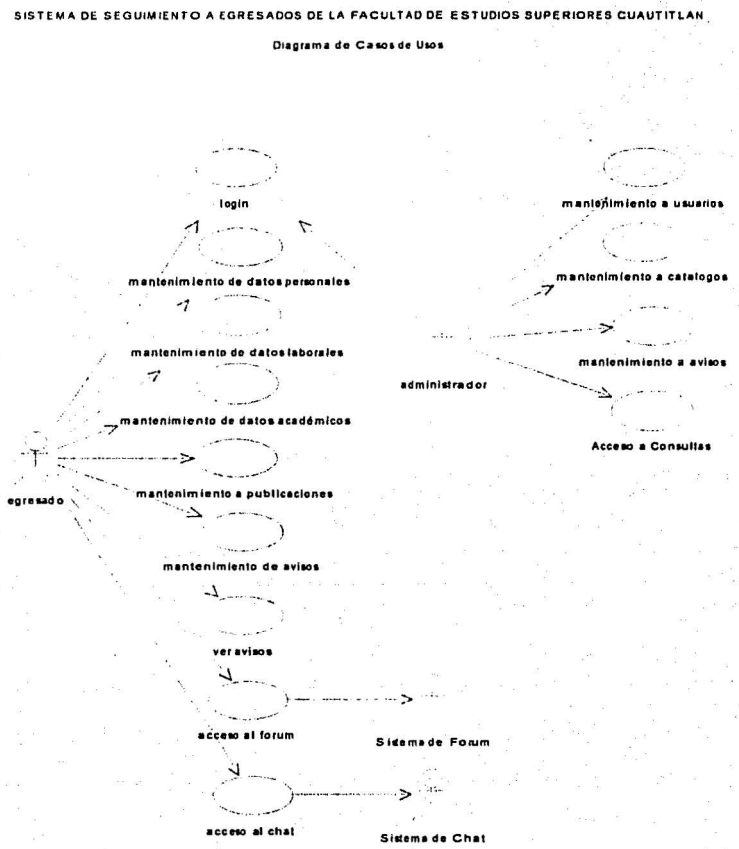
En este caso se tomo la decisión de utilizar la definición de tablas, como una alternativa al Diccionario de datos.

El modelo de datos que se diseño para la creación de la Base de Datos que utilizará el sistema, cumple hasta la 3ra. Forma Normal establecida para el modelo Entidad-Relación.

La base de datos que fue implementada bajo MySQL, nos dimos cuenta que a pesar de que se llego a la 3ra. Forma Normal, permitía en algún momento que se pudiera romper la integridad de la información; sin que esto llegará a afectar a todo el diseño.

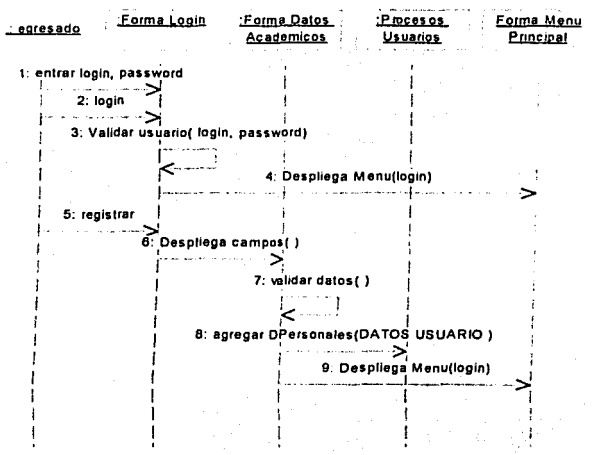
Esto es debido a que MySQL no es tan rígida la validación al realizar algún proceso en la base de datos, esto se puede prevenir por medio de la programación.

1. Diagrama de Casos de Uso



2. Diagramas de Secuencias

2.1 Acceso al Sistema

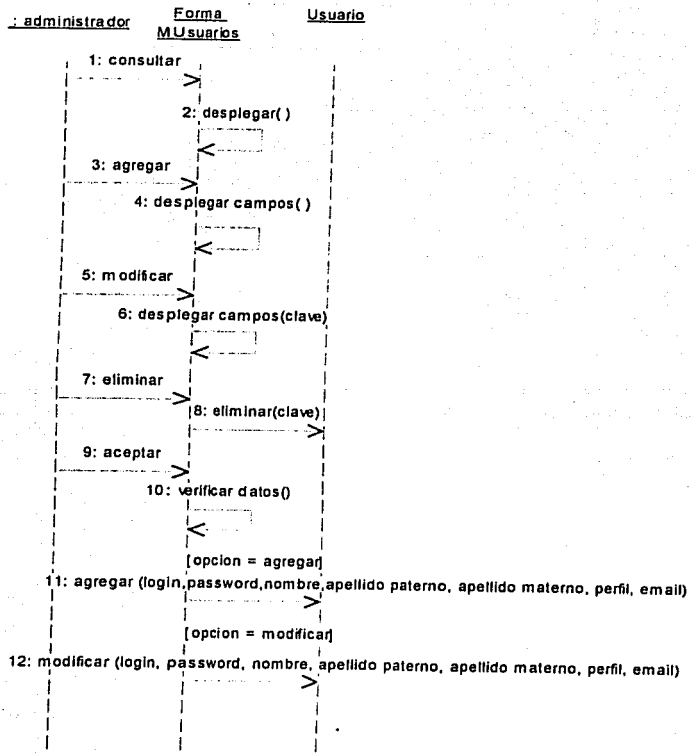


DATOS USUARIO

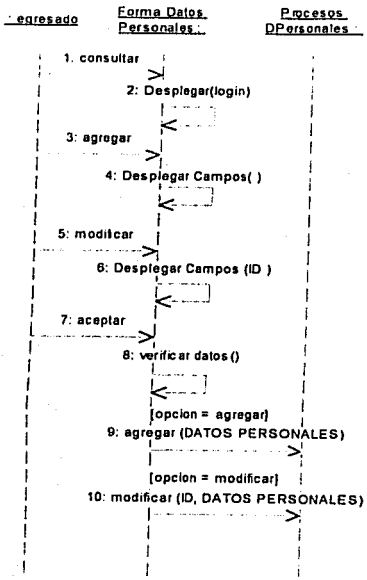
login
password

numero de cuenta
nombre
apellido paterno
apellido materno
calle
colonia
numero interior
numero exterior
municipio
ciudad
cp
telefono
email

2.2 Mantenimiento de Usuarios

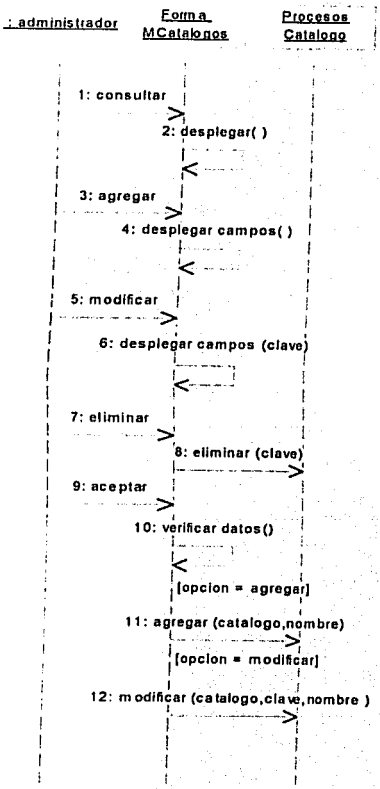


2.3 Mantenimiento de Datos Personales



DATOS PERSONALES:
numero de cuenta
nombre
apellido paterno
apellido materno
calle
colonia
numero interior
numero exterior
municipio
ciudad
cp
telefono
email
generacion

2.4 Mantenimiento de Catálogos



Los tipos de Catalogos incluye los siguientes datos:

ESCUELAS:
clave
nombre

NIVELES:
clave
nombre

AREAS:
clave
nombre

GENERACIONES:
clave
nombre
descripcion

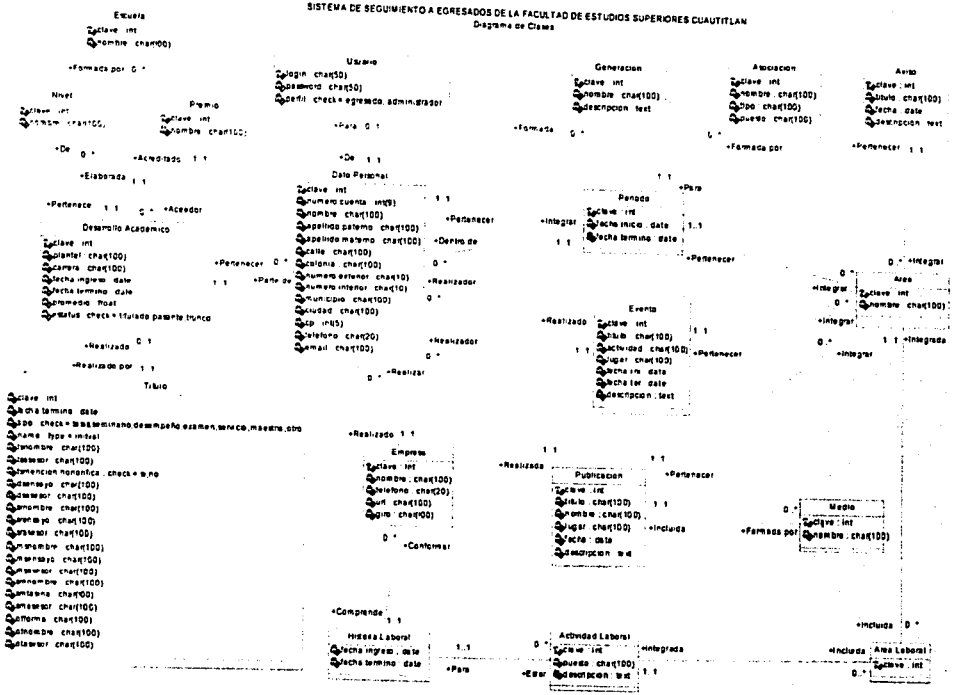
MEDIOS:
clave
nombre

Falta

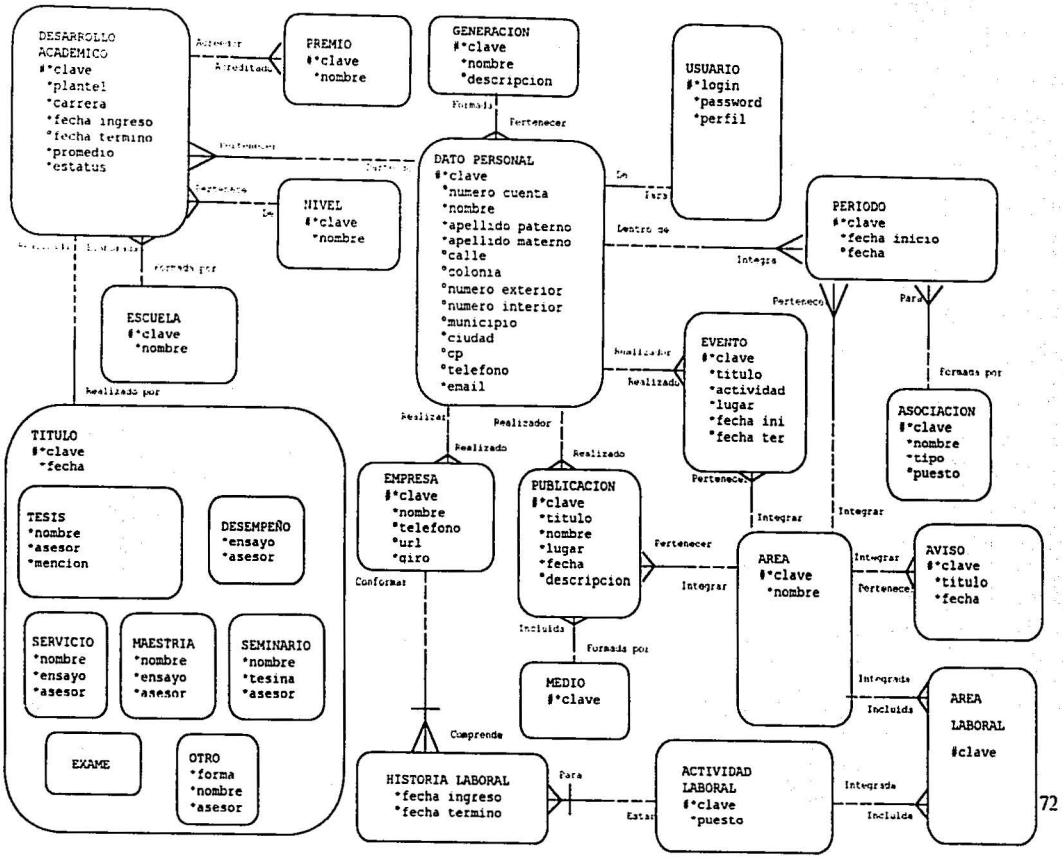
Página

170

3. Diagrama de Clases



4. Modelo Entidad Relación



5. Definición de Tablas

USUARIOS: Contiene información de los usuarios que se registran en el sistema.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
login	Primaria	Varchar	50	No nulo / único			
password		Varchar	50	No nulo			
perfil		Enum		No nulo	egresado, administrador		

GENERACIONES: Contiene información de las generaciones que han egresado de la Licenciatura.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo / único			
descripcion		Text		Nulo			

DATOS_PERSONALES: Contiene información personal de los usuarios registrados.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
numero_cuenta		Int zerofill	9	Nulo			
nombre		Varchar	100	No nulo			
apellido_paterno		Varchar	100	No nulo			
apellido_materno		Varchar	100	No nulo			
calle		Varchar	100	Nulo			
colonia		Varchar	100	Nulo			
numero_exterior		Varchar	10	Nulo			
numero_interior		Varchar	10	Nulo			
municipio		Varchar	100	Nulo			
ciudad		Varchar	100	Nulo			
cp		Int zerofill	5	Nulo			
telefono		Varchar	20	Nulo			
email		Varchar	100	No Nulo			
us_login	Foránea	Varchar	50	No nulo		usuarios	login
ge_clave	Foránea	Int		Nulo		generaciones	clave

ESCUELAS: Contiene información de diversas escuelas.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo / único			

NIVELES: Contiene información de los niveles de estudio.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo / único			

DESARROLLOS_ACADÉMICOS: Contiene información referente a su trayectoria académica de cada usuario.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
plantel		Varchar	100	No nulo			
carrera		Varchar	100	No nulo			
fecha_ingreso		Date		No nulo			
fecha_termino		Date		Nulo			
promedio		Float		No nulo			
estatus		Enum		No nulo	titulado, pasante, trunco		
dp_clave	Foránea	Int		No nulo		datos_personales	clave
ni_clave	Foránea	Int		No nulo		niveles	clave
es_clave	Foránea	Int		No nulo		escuelas	clave

PREMIOS: Contiene información de los premios obtenidos en su trayectoria académica.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo			
da_clave	Foránea	Int		No nulo		desarrollos_academicos	clave

TITULOS: Contiene información de las formas de titulación.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
fecha_termino		Date		No nulo			
tipo		Enum		No nulo	tesis, seminario, desempeño, examen, servicio, maestria, otro		
tsnombre		Varchar	100	Nulo			
tsasesor		Varchar	100	Nulo			
tsmencion_honorifica		Enum		Nulo	si, no		
dsensayo		Varchar	100	Nulo			
dsasesor		Varchar	100	Nulo			
srnombre		Varchar	100	Nulo			
srensayo		Varchar	100	Nulo			
srasesor		Varchar	100	Nulo			
msnombre		Varchar	100	Nulo			
msensayo		Varchar	100	Nulo			
msasesor		Varchar	100	Nulo			
smnombre		Varchar	100	Nulo			
smtesina		Varchar	100	Nulo			
smasesor		Varchar	100	Nulo			
otforma		Varchar	100	Nulo			
otnombre		Varchar	100	Nulo			
otasesor		Varchar	100	Nulo			
da_clave		Int		No nulo		desarrollo_academico	clave

ACTIVIDADES_LABORALES: Contiene información de la actividad laboral que ha realizado el usuario.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
puesto		Varchar	100	No nulo			
descripcion		Text		Nulo			

EMPRESAS: Contiene información de empresas relacionadas a la actividad laboral.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo			
telefono		Varchar	20	Nulo			
url		Varchar	100	Nulo			
giro		Varchar	100	No nulo			
dp_clave	Foránea	Int		No nulo		datos_personales	clave

HISTORIAS_LABORALES: Contiene información del periodo que abarca cada actividad laboral.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
fecha_ingreso		Date		No nulo			
fecha_termino		Date		Nulo			
al_clave	Primaria	Int		No nulo / único		actividades_laborales	clave
em_clave	Primaria	Int		No nulo / único		empresas	clave

AREAS: Contiene información de las diversas áreas que conforman la Licenciatura I

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo / único			

AREAS_LABORALES: Contiene información de las áreas en que se desempeña la actividad laboral.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
al_clave	Foránea	Int		No nulo		actividades_laborales	Clave
ar_clave	Foránea	Int		No nulo		areas	Clave

MEDIOS: Contiene información de los diversos medios de publicación.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo / único			

PUBLICACIONES: Contiene información de las publicaciones realizadas por los usuario

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
titulo		Varchar	100	No nulo			
nombre		Varchar	100	No nulo			
lugar		Varchar	100	No nulo			
fecha		Date		No nulo			
descripcion		Text		Nulo			
ar_clave	Foránea	Int		No nulo		areas	clave
dp_clave	Foránea	Int		No nulo		datos_personales	clave
me_clave	Foránea	Int		No nulo		medios	clave

AVISOS: Contiene información de los avisos que registra el administrador.

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
titulo		Varchar	100	No nulo			
fecha		Date		No nulo			
descripcion		Text		Nulo			
ar_clave	Foránea	Int		No nulo		areas	clave

EVENTOS: Contiene información de los eventos que ha participado el usuario

Columna	Tipo de llave	Tipo de Dato	Longitud	Nulo / Único	Val. por Default	Tabla de Ref.	Col. de Ref.
clave	Primaria	Int		No nulo / único			
titulo		Varchar	100	No nulo			
actividad		Varchar	100	No nulo			
lugar		Varchar	100	No nulo			
fecha_ini		Date		No nulo			
fecha_ter		Date		Nulo			
descripcion		Text		Nulo			
ar_clave	Foránea	Int		No nulo		areas	clave
dp_clave	Foránea	Int		No nulo		datos_personales	clave

ASOCIACIONES: Contiene información de las asociaciones a las que ha pertenecido el usuario.							
Columna	**Tipo de llave**	**Tipo de Dato**	**Longitud**	**Nulo / Único**	**Val. por Default**	**Tabla de Ref.**	**Col. de Ref.**
clave	Primaria	Int		No nulo / único			
nombre		Varchar	100	No nulo			
tipo		Varchar	100	No nulo			
puesto		Varchar	100	Nulo			

PERIODOS: Contiene información del periodo que ha pertenecido a una asociación.							
Columna	**Tipo de llave**	**Tipo de Dato**	**Longitud**	**Nulo / Único**	**Val. por Default**	**Tabla de Ref.**	**Col. de Ref.**
Clave	Primaria	Int		No nulo / único			
fecha_inicio		Date		No nulo			
fecha_termino		Date		Nulo			
ar_clave	Foránea	Int		No nulo		areas	clave
dp_clave	Foránea	Int		No nulo		datos_personales	clave
as_clave	Foránea	Int		No nulo		asociaciones	clave

b) DISEÑO

El proceso de diseño del sistema que se implemento fue el modelo en cascada modificado, debido a que el proceso que se sigue es iterativo, y nos permitió regresar a etapas anteriores en caso de que se requiera, sin que esto implique que los demás procesos sigan su curso.

El diseño esta conformado por las siguientes partes:

Parte de Egresados lo conforman las siguientes pantallas:

- Datos Personales.
- Datos Académicos.
- Datos Laborales.
- Publicaciones.
- Asociaciones.
- Eventos.

Existen otras aplicaciones que son complementarias al sistema

Parte de Administración lo conforman las siguientes pantallas:

- Catálogos
    - Usuarios
    - Escuelas
    - Niveles
    - Generaciones
    - Medios
- Avisos
    - Por número de cuenta
    - Por nombre

- Generaciones
- Niveles
- Titulados
- Forma de Titulación
- Premios
- Mención Honorífica
- Actividad laboral por área
- Actividad laboral actual
- Puesto
- Experiencia laboral
- Publicaciones
- Eventos
- Asociaciones

La definición de todas las pantallas, se describe en el diseño de pantallas.

1. DISEÑO DE PANTALLAS

1. EGRESADOS

Dentro del Sistema de Seguimiento a Egresados, existe una aplicación que conforma todo lo referente a los datos de un egresado, por lo que se requiere de una pantalla que permita el acceso al sistema.

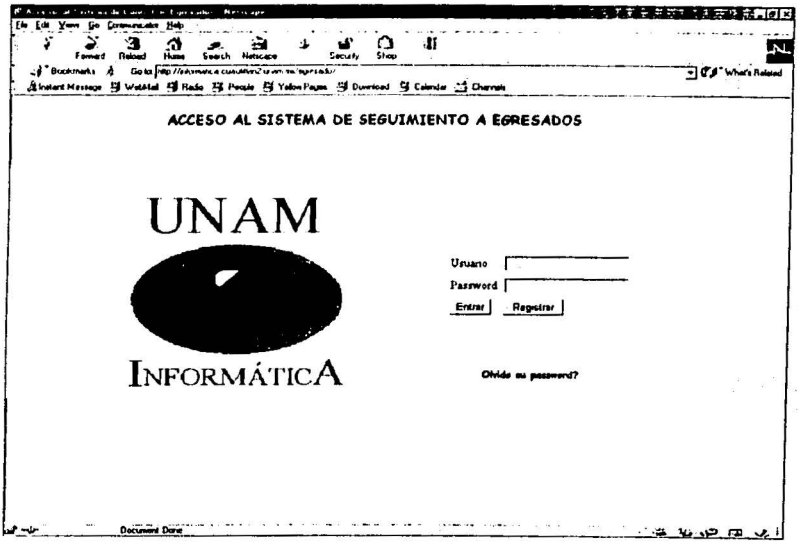


Imagen 1 "Acceso al Sistema de Seguimiento a Egresados"

Esta pantalla permite al egresado poder ingresar un usuario y un password que permitirá dar acceso al Sistema, si el usuario no cuenta con la información correspondiente, es necesario el registro, por lo que se requiere de una pantalla en la que el egresado pueda ingresar su información.

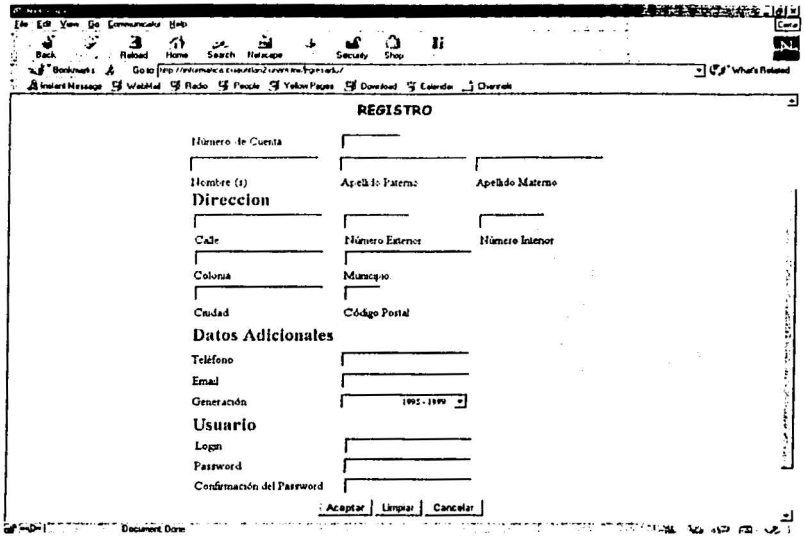


Imagen 2 "Registro"

Esta pantalla, permite al egresado ingresar sus datos personales. La pantalla de registro esta formado por el Número de Cuenta, el Nombre, la Dirección, Datos Adicionales que son útiles para la comunicación con el egresado, por lo que se requiere el teléfono y su email, así como conocer la generación a la que pertenece el usuario, por último los datos para conectarse al sistema, en este caso se requiere de un login, un password y una confirmación del password, una vez que se hayan ingresado todos los datos requeridos, se almacenarán en el Sistema y presentará la pantalla de la imagen 1.

Una vez que el sistema verifica al usuario, el sistema presentará el menú de las opciones que podrá utilizar para ingresar su información.

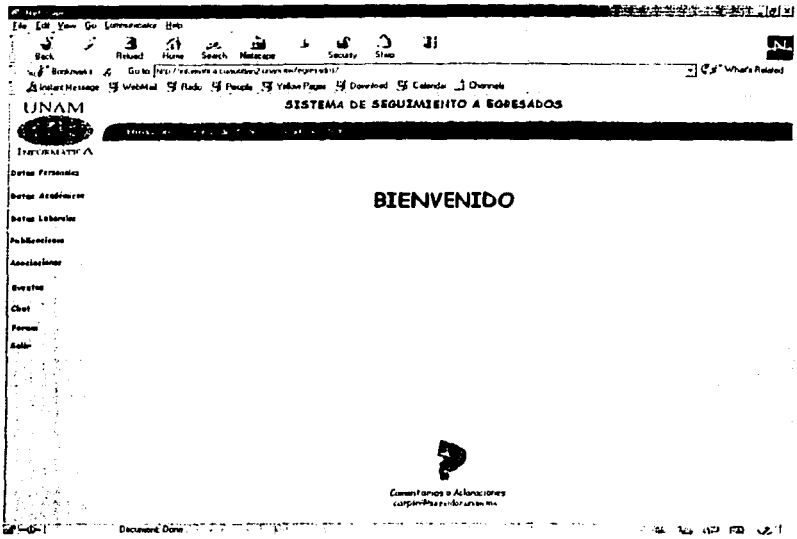


Imagen 3 "Menú del Sistema para los egresados"

Esta pantalla esta formada por:

Datos Personales. Esta opción contiene la información personal del usuario, la cual fue ingresada cuando el usuario fue registrado. La información de sus datos personales, pueden ser actualizados, así como su password de ingreso.

Datos Académicos. Esta opción contiene la información de su desarrollo académico a partir de la Licenciatura, así como los premios que haya obtenido. La información almacenada puede ser insertada, modificada, consultada o borrada.

Datos Laborales. Esta opción contiene la información de sus actividades laborales, iniciando

con las empresas en las cuales haya laborado, además de las actividades realizadas en cada una de ellas. La información almacenada puede ser insertada, modificada, consultada o borrada.

Publicaciones. Esta opción contiene las información de las publicadas realizadas. La información almacenada puede ser insertada, modificada, consultada o borrada.

Asociaciones. Esta opción contiene las información de las asociaciones a las que pertenece o haya pertenecido. La información almacenada puede ser insertada, modificada, consultada o borrada.

Eventos. Esta opción contiene la información de los eventos en los que haya participado. La información almacenada puede ser insertada, modificada, consultada o borrada.

Chat. Esta opción permite a los egresados poder comunicarse con otro egresado.

Forum. Esta opción permite a los egresados realizar debates sobre algún tema.

Salir. Esta opción permite al egresado cerrar el Sistema.

2. ADMINISTRADOR

Dentro del Sistema de Seguimiento a Egresados, existe una aplicación que permite a una persona encargada del sistema ser el administrador, por lo que se requiere de una pantalla que permita el acceso a la administración del sistema.

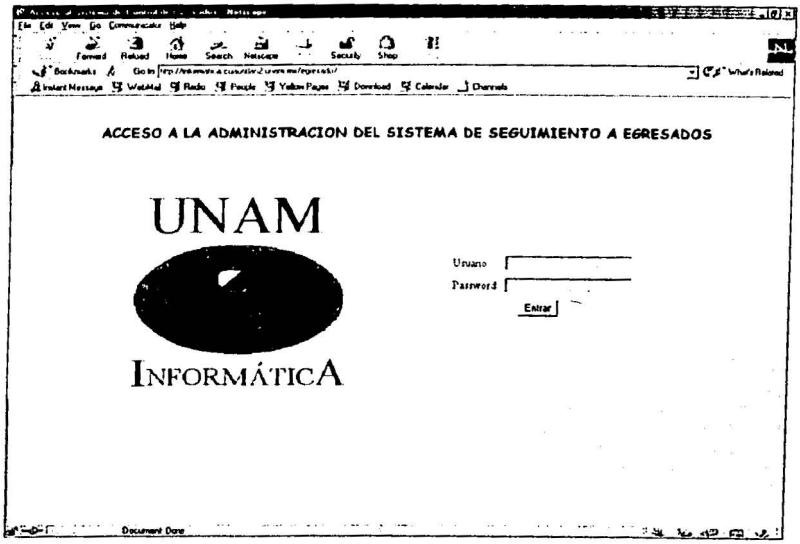


Imagen 4 "Acceso al Sistema de Administración"

Esta pantalla permite al administrador poder ingresar un usuario y un password que permitirá dar acceso al Sistema de Administración.

Una vez ingresado al sistema, el Sistema de administración presentará un menú con las opciones requeridas para administrar el sistemas.

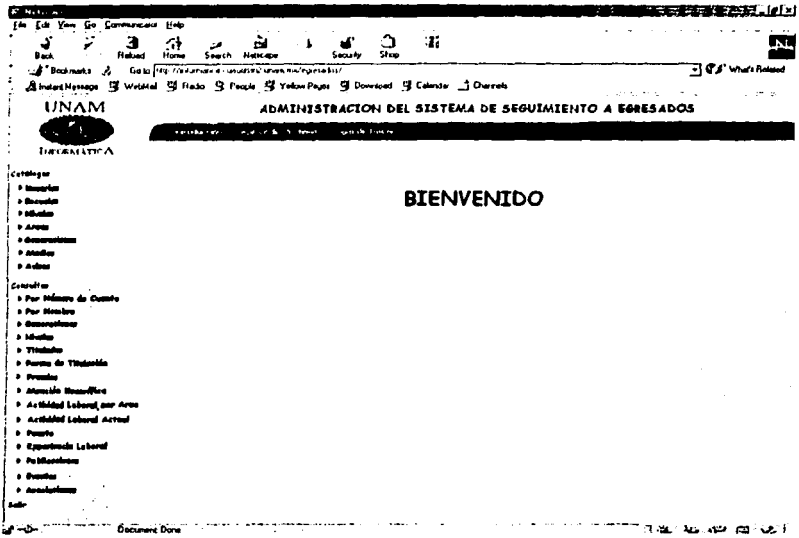


Imagen 5 "Menú de Administración del Sistema"

Esta pantalla, esta dividida en Catálogos, Consultas y Salir.

Catálogos. La opción de catálogos esta formada por toda aquella información que el egresado requiere para poder ingresar su información. Los catálogos son:

- Usuarios: Este catálogo tiene registrado todos los usuarios que integran el sistema, dentro de este catálogo se pueden registrar usuarios, modificarlos, eliminarlos y consultarlos, tanto para egresados como para administradores, en esta opción se realiza el cambio del password para el usuario admin..
- Escuelas. Este catálogo tiene registrado todas las escuelas a las que han pertenecido

los egresados. La información almacenada puede ser insertada, modificada, consultada o borrada, también puede ser insertada dentro de la forma de Datos Académicos.

- Niveles. Este catálogo tiene registrado los niveles escolares que hayan estudiado los egresados. La información almacenada puede ser insertada, modificada, consultada o borrada.
- Generaciones. Este catálogo tiene registrado las generaciones participantes en la Licenciatura. La información almacenada puede ser insertada, modificada, consultada o borrada.
- Medios. Este catálogo tiene registrado los medios en donde se realizaron publicaciones. La información almacenada puede ser insertada, modificada, consultada o borrada.
- Avisos. Este catálogo tiene registrado los avisos que se realicen para los egresados. La información almacenada puede ser insertada, modificada, consultada o borrada.

Consultas. Las consultas permiten al administrador tener conocimiento sobre la información de interés para tener un mejor control del sistema, esta formada por las siguientes consultas.

- Por número de cuenta. Esta consulta despliega la información de un egresado por su número de cuenta.
- Por nombre. Esta consulta despliega la información de un egresado por su nombre, apellido paterno y apellido materno. Se pueden realizar consultas intercalando esta información.

- Generaciones. Esta consulta despliega la información de todos los egresados pertenecientes a la generación seleccionada.
- Niveles. Esta consulta despliega la información de todos los egresados pertenecientes al nivel seleccionado.
- Titulados. Esta consulta despliega la información de todos los egresados que estén titulados.
- Forma de Titulación. Esta consulta despliega la información de todos los egresados dependiendo en primer termino del nivel, para después seleccionar la forma de titulación.
- Premios. Esta consulta despliega la información de todos los egresados que hayan obtenido un premio.
- Mención Honorífica. Esta consulta despliega la información de todos los egresados que hayan obtenido Mención Honorífica en su tesis.
- Actividad laboral por área. Esta consulta despliega la información de todos los egresados que hayan realizado alguna actividad laboral en el área seleccionada.
- Actividad laboral actual. Esta consulta despliega la información de todos los egresados que estén actualmente realizado alguna actividad laboral en el área seleccionada.
- Puesto. Esta consulta despliega la información de todos los egresados que hayan tenido algún puesto a consultar.
- Experiencia laboral. Esta consulta despliega la información de todos los egresados que hayan tenido experiencia en la actividad a consultar.

- Publicaciones. Esta consulta despliega la información de todos los egresados que hayan realizado alguna publicación.
- Eventos. Esta consulta despliega la información de todos los egresados que hayan participado en algún evento.
- Asociaciones. Esta consulta despliega la información de todos los egresados que pertenezcan a una asociación.

Salir. Esta opción permite al administrador cerrar el Sistema.

c). DESARROLLO

El desarrollo del Sistema de Seguimiento a Egresados, es una aplicación para internet, utilizando un navegador como cliente como servidor de bases de datos actuará el servidor de web Apache con PHP como la aplicación de desarrollo y conexión a base de datos.

El sistema estará compuesto por cinco elementos principales:

- GNU/Linux como sistema operativo abierto
- Apache como servidor de web
- PHP como módulo de ampliación de Apache para acceder a la base de datos
- MySQL como base de datos
- Una interfaz web a MySQL, la cual fue realizada con la integración de PHP, HTML y JavaScript

Todos los elementos del sistema se obtuvieron sin ningún tipo de costo por lo que el montaje inicial del sistema y su uso no conllevaron a ninguna inversión.

d). IMPLEMENTACION

Una vez terminado el desarrollo, así como las pruebas de las aplicaciones creadas, se entra a la última fase del sistema, la implementación.

La implementación se realizó en un Servidor que se encuentra dentro de la Licenciatura en Informática, este servidor tiene instalada la versión Linux Mandrake 8.0, el cual tiene configurado el Servidor Web Apache.

Se configuró el ambiente de desarrollo, por lo que se instaló el PHP y el manejador de Base de Datos Mysql, una vez instaladas las aplicaciones, se realizó la compilación de los archivos de configuración para la conexión de PHP hacia Mysql y de Apache a PHP.

Al ver que todo esta funcionando, se dio paso a la creación de la Base de Datos, así como un usuario que tenga los privilegios requeridos para realizar las operaciones correspondientes a cada pantalla, una vez terminada se corrio el script que generó todas las tablas, e índices que forman la base de datos.

Se crearon los directorios necesarios dentro del Servidor de Linux. Por último se realizó una prueba con cada una de las aplicaciones para ver que todo funcione correctamente.