UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE PORTALES DE BIBLIOTECAS DIGITALES

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

ALEJANDRO CRUZ SANTOS
EDUARDO HERNANDEZ BELMONT
GINA AIDE PEREZ JUAREZ

DIRECTOR DE TESIS: ING. ALBERTO GONZALEZ GUIZAR

CIUDAD UNIVERSITARIA, JUNIO DEL 2005

AGRADECIMIENTOS

A mi único Amor:

Yeshúa ha Mashiaj

Gracias mi dulce Yeshúa Adonáy por darme la vida, por otorgarme la oportunidad de comenzar de nuevo, te agradezco por que eres bondadoso y me permitiste experimentar en aquellos sueños cosas de ti que nunca imagine vivir, gracias por que en todo momento estas conmigo, gracias por que me consuelas compartiendo de tu fragancia y de tu presencia haciéndome sentir que estas Vivo y que puedo sentirte a pesar de mi.

Gracias por todas aquellas personas que me han apoyado, por mis profesores, mis padres, mis amigos y por que nos guardas en tus bellas manos y bajo la sombra de tus alas estamos seguros.

Quiero agradecerte por que este sueño lo hiciste realidad.

Nunca olvidare lo que has hecho por mí, eres mi todo, eres mi único Dios, mi Salvador y mi Rey, mucho tiempo estuve buscando alguien que me Amara de verdad, que sanara mis heridas, me llenara de paz y colmara mi corazón de su Amor y me encontraste al ver mi condición, eres fiel Adonáy, eres lo más hermoso que me ha pasado en la vida, no quiero perderte Eloháy por que se que si no estas conmigo mi vida no vale nada y no quiero vivir sin ti.

Bendice a mi Familia, Amigos, a mis Profesores y también a sus familias.

Bendito sea el nombre de tu Reino, Glorioso por siempre Jamás. Rey de Gloria, de Honra, de Majestad… El todo Poderoso; Yeshúa es tu nombre… Eres el más hermoso de los hijos de los hombres, por eso el Padre te ha coronado de Gloria, de Honra y de Honor, Baruj Ata Adonáy por siempre y siempre.

¡ ¡ ¡ Agi oje vet mi Adonáy y Mashiaj, Melej y Eloháy ! ! !
Kadosh, Kadosh, Kadosh Adonáy Elohim Tsebaot!!!

Con Amor……

Gina Aide Perez Juárez.

A mis padres:

Isabel Juárez y Alfredo Pérez: *Gracias por todo su apoyo, sus consejos, su paciencia, su Cariño, por darme ánimo de seguir adelante, por enseñarme a fijar metas y cumplirlas. Los Quiero Mucho, gracias por todos aquellos valores que me han inculcado para mi crecimiento tanto como para ser una mejor persona, como para ser una mejor mujer cada día.*

A mis Hermanas:

Dennys y Gaby Pérez: *Por todo su apoyo, su tiempo, sus consejos y su Cariño. Las quiero Mucho!!!*

A mi Tio:

Francisco Pérez: *Mi querido Tio Panchito, quiero expresar mi gratitud hacia ti por todo tu apoyo, tus apapachos, tu preocupación para que siga adelante, gracias por que cuando más necesite de alguien estabas conmigo.. Solo puedo decirte que... TE QUIERO MUCHO!!!*

A mi Amiga y Hermanita:

Ingrid Colín (Muñeca): *Por todo tu cariño, tus consejos, tu confianza, tu preocupación y por todas aquellas oraciones que elevaste a Papito Dios para que me ayudara y estuviera conmigo, gracias por ser mi Amiga y mi Hermanita. Te quiero mucho mi Muñequita querida y que Dios te siga bendiciendo!!!*

A mi Hermanita y amiga:

Mariana Franco (Marianita): *Por tu apoyo, tu tiempo y por todas tus oraciones a El-Elyon para mi, por tu preocupación y por siempre estar dispuestas a escucharme. Muchas Gracias...Hasem sea contigo y con el dulce Danielito. Los quiero mucho.*

A mi Director de Tesis y Amigo:

Alberto Guizar: *Beto gracias por tu paciencia, tiempo, dedicación, consejos y por preocuparte por mi, gracias por todo tu apoyo para ayudarnos a terminar este anhelado sueño. T.Q.M!!!*

A mi Profesora:

Lucila Patricia Arellano: *Profesora muchas gracias por todo su Apoyo, por preocuparse por mi, por el tiempo que me dedico y por que cuando más necesite de apoyo usted me tendió su mano para ayudarme, usted es para mi excelente profesora y una excelente Amiga. Reciba de mi parte: Mi Cariño y Todo mi Respeto. Dios la Bendiga a usted y a su Hermosa Familia!!!*

A mis Amigos:

Victor Bautista, Jorge Palomares, Felipe López, Nancy Cortés, Julio Bernal, Santiago (mi Master) *muchas gracias por su apoyo, consejos, por ayudarme en este sueño, pero sobre todo muchas gracias por su Cariño. Los quiero muchote!!!*

*A mis **Profesores**, a mi **Facultad de Ingeniería** y sobre todo a mi querida y amada **Universidad Nacional Autónoma de México** gracias por la oportunidad que me brindaron, gracias por toda la paciencia y la dedicación, por la enseñanza recibida, y su tiempo.....Muchas Gracias!!!*

Gina Aide Pérez Juárez.

A MIS PADRES:

MACARIO MANUEL CRUZ SANTIAGO
MA. DEL ROSARIO SANTOS MONTESINOS

Mil gracias por su apoyo, sus consejos y enseñarme que con humildad, respeto y disciplina se logra alcanzar una meta. Gracias por ser el mejor ejemplo que he podido tener.

A MIS HERMANOS:

MANUEL Y ELSA BEATRIZ CRUZ SANTOS

Por esforzarnos a seguir mejorando en nuestra formación profesional. Gracias por preocuparse y estar ahí para ayudarme.

A MIS TIOS:

SALVADOR SANTOS MONTESINOS
TRINIDAD CRUZ LOPEZ

Porque ante las adversidades siempre han brindado su apoyo incondicional en todo momento, gracias.

CON CARIÑO:

Ing. Sandra L. Pioquinto Gutiérrez por todos estos años de amistad y por apoyarme desde que nos conocimos, te agradezco los momentos gratos y sinceramente hiciste que el tiempo de clases se hiciera más agradable.

A MI ASESOR DE TESIS:

ING. ALBERTO GONZALEZ GUIZAR

Por su amabilidad, observaciones, apreciaciones, recomendaciones y paciencia para la culminación de esta tesis.

A mis compañeros, profesores, a la Facultad de ingeniería, y sobre todo a la UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO por forjar mi educación con valores y conocimientos, gracias.

Alejandro Cruz Santos.

Gracias amado Señor.

Gracias por el don de la vida y por las maravillosas mujeres que me permitiste tener por madre y abuelita, gracias por darles la fuerza necesaria y la sabiduría para educar a sus hijos, lo mismo te agradezco por mi padre por cuanto ama la vida y darme en herencia sus sabios consejos. Gracias por permitirles dar el ejemplo digno de disciplina y amor a su familia.

Gracias por permitirme ser millonario al brindarme la familia que tengo, gracias por todos ellos y por los que vendrán.

Gracias por Claudia, por ese sentimiento tan bonito que tiene de amar y ser mi acompañante en parte de esta travesía.

Gracias por darme de beber de tus manos cuando tuve sed en el desierto, gracias por abrazarme y tomar mi mano cuando todo era apariencia, gracias por ser mi camino y mi vida.

Gracias por todos aquellos héroes anónimos que diariamente desde muy temprano con sus enseñanzas, enaltecen el nombre de México y de esta honorable Universidad, de la que me siento inmensamente orgulloso y eternamente agradecido por el privilegio brindado de pertenecer a ella.

Desde aquí les rindo un homenaje con el más humilde corazón y con el más grande respeto, a quienes siempre llevo en mi mente y cada paso que doy es dedicado en silencio a su memoria, cariño y esfuerzo. Sepan que su amor, dedicación y empeño depositado, no ha sido en vano y espero algún día poder hacerlos sentir tan orgullosos como hoy me siento de todos ustedes.

Eduardo Hernández Belmont.

Índice

INTRODUCCION

Actualmente en México se está observando la gran importancia que tiene la información digital en la vida cotidiana, esto ha permitido que la demanda de esta información vaya aumentando y por lo tanto se tenga que hacer uso de sistemas sofisticados para lograr la generación, almacenamiento, administración y acceso a este tipo de datos.

Para poder realizar dichas actividades ha sido preocupación de la Universidad Nacional Autónoma de México y de la Dirección General de Bibliotecas en particular, utilizar las tecnologías disponibles en beneficio de la universalización del acceso a la información.

En especial, han sido muchos los estudios realizados para que la comunidad académica pueda consultar fácilmente las publicaciones electrónicas almacenadas y catalogadas en todas las bibliotecas de la Universidad.

Existen ciertos acervos bibliotecarios que no son accesibles o de conocimiento para cierto sector de la comunidad universitaria y público en general, actualmente dicha información no se encuentra clasificada y ordenada para ser consultada.

Por ello la necesidad de implementar una base de datos y una interfaz interactiva de consulta en una aplicación de Internet que permita visualizar fácilmente la información capaz de satisfacer las necesidades de búsqueda de información de los usuarios.

Actualmente no existe un sistema con estas características dentro de la mayoría de las Facultades y que sea capaz de brindar el servicio adecuado de consulta para la comunidad universitaria.

El sistema de consulta de material no librario tendrá en definitiva una mayor ayuda sobre toda aquella persona que realice una investigación entorno a un tópico muy particular, dará una mayor visión para poder discernir entre los diferentes materiales que se encuentran almacenados en la Universidad Nacional Autónoma de México.

Por esta razón creemos que con este paso de difundir a través del Web el acervo de material no librario de estas primeras 2 bibliotecas, estamos concretando una meta largamente acariciada.

Esperamos como resultados al realizar dicho portal obtener una mayor eficiencia en la búsqueda de la información complementando de esta manera cualquier tipo de investigación en forma sencilla, dinámica y agradable para el usuario.

Obtener un sistema de bases de datos en el cual se encontrarán almacenados todos los acervos de material no librario de algunas de las bibliotecas de la UNAM al alcance del público en general.

Pretendemos establecer un sistema que pueda ser modificado de acuerdo a las necesidades presentadas en un futuro y obtener del sistema la recuperación de información que sea de utilidad para los usuarios.

CAPÍTULO 1

ANTECEDENTES

1.1 BIBLIOTECA DIGITAL.

Biblioteca digital, es un repositorio de acervos y contenidos digitalizados, almacenados en diferentes formatos electrónicos por lo que el original en papel, en caso de existir, pierde supremacía. Generalmente, son bibliotecas pequeñas y especializadas, con colecciones limitadas a sólo algunos temas.

"Las Bibliotecas Digitales son organizaciones que proveen los recursos, incluyendo personal especializado, para seleccionar, estructurar, distribuir, controlar el acceso, conservar la integridad y asegurar la persistencia a través del tiempo de colecciones de trabajos digitales que estén fácil y económicamente disponibles para usarse por una comunidad definida o para un conjunto de comunidades."

Características.

La forma de estructurar sus servicios, está relacionada con el diseño de interfaces para operar vía sistemas de telecomunicación.

La presencia de servicios en donde existe una separación física entre el bibliotecario y los usuarios, gran parte o todo el tiempo, durante el proceso de la prestación de servicios.

La posibilidad de enlace entre diversos sistemas de servicios bibliotecarios y de información documental vía telecomunicaciones.

La creación de nuevos medios para clasificar y formar colecciones de documentos digitales destinados a instrumentos para navegación y consulta de los contenidos de los documentos del acervo de la biblioteca.

El diseño, organización y presentación de servicios en donde los conocimientos relativos a las tecnologías de la información y las telecomunicaciones son un componente esencial para la construcción y operación de la biblioteca.

Transferencia de documentos digitales vía telecomunicaciones, que permiten la disponibilidad inmediata del documento, pero también obligan a reconsiderar la idea del préstamo interbibliotecario; en el sentido de conciliar los derechos e intereses de autores y editores, con la necesidad de facilitar la libre circulación pública de documentos e información.

Diseño de sistemas de administración flexibles orientados a la gestión de la información para facilitar el acceso y disponibilidad de datos e información específica.

El auge de las bibliotecas digitales está motivado por la facilidad de acceso a una gran variedad de información, ya que éstas extienden los servicios de las bibliotecas convencionales [Sánchez 1997]. Desde los inicios de las bibliotecas convencionales, los usuarios de las mismas han necesitado de auxiliares para que la información almacenada pueda transformarse en conocimiento y sea útil.

CAPÍTULO 2

CONCEPTOS BASICOS

2.1 Bases de Datos.

Las computadoras han evolucionado mucho las tareas del ser humano, ya que simplifican, facilitan y organizan grandes cantidades de información, esto se realiza mucho dentro de las industrias, servicios médicos, instituciones financieras y otras instituciones, por esta razón, debe existir un lugar en donde se pueda guardar toda la información que esta manejando para poderla consultar nuevamente en determinado momento, a este lugar se le llama "Bases de Datos".

2.1.1 Definición.

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

Una *base de datos* es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización.

En una base de datos, además de los datos, también se almacena su descripción. La base de datos es un gran almacén de datos que se define una sola vez y que se utiliza al mismo tiempo por muchos departamentos y usuarios.

Además, la base de datos no sólo contiene los datos de la organización, también almacena una descripción de dichos datos.

**Una Base de Datos es un conjunto auto descriptivo de registros integrados. Es auto descriptivo porque contiene una descripción de si misma en un diccionario de datos, el cual también se conoce como directorio de datos o meta datos.

Una Base de Datos es un conjunto de registros integrados porque la relación entre estos se almacena en la base.

2.1.2 Tipos de usuarios.

Podemos definir a los usuarios como toda persona que tenga todo tipo de contacto con el sistema de base de datos desde que este se diseña, desarrolla e implementa.

En primer lugar, los usuarios finales, que hacen un uso limitado de las capacidades del sistema, normalmente referentes a introducción, manipulación y consulta de datos. Los usuarios finales pueden ser especializados o principiantes, dependiendo de su nivel de interacción con el sistema.

Usuarios especializados.

Algunos usuarios especializados escriben aplicaciones de base de datos especializadas que no encajan en el marco tradicional de procesamiento de datos.

Usuarios principiantes.

Estos usuarios interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente en el sistema de base de datos, podemos mencionar al usuario ingenuo como el usuario final que utiliza el sistema de base de datos sin saber nada del diseño interno del mismo.

En segundo lugar hay que citar a los programadores de base de datos, encargados de escribir aplicaciones limitadas, mediante el lenguaje de programación facilitado por el SGBD, que faciliten la ejecución de tareas por parte de los usuarios finales.

Programadores de aplicaciones.

Los programadores que interactúan con el sistema por medio de llamadas en DML (Lenguaje de Manipulación de Datos), las cuales están incorporadas en un programa escrito en un lenguaje de programación (Por ejemplo, COBOL, Pascal, C, etc.).

DBA.

Por último, el administrador de base de datos (DBA, Data Base Administrator) cumple las importantes funciones de crear y almacenar las estructuras de la base de datos, definir las estrategias de respaldo y recuperación, vincularse con los usuarios y responder a sus cambios de requerimientos, y definir los controles de autorización y los procedimientos de validación.

2.1.3 Administrador de base de datos (DBA).

Es la persona o equipo de personas profesionales responsables del control y manejo del sistema de base de datos, generalmente tiene(n) experiencia en DBMS, Diseño de bases de datos, Sistemas operativos, Comunicación de datos, Hardware y Programación.

Los sistemas de base de datos se diseñan para manejar grandes cantidades de información, la manipulación de los datos involucra tanto la definición de estructuras para el almacenamiento de la información como la provisión de mecanismos para la manipulación de la información, además, un sistema de base de datos debe de tener implementados mecanismos de seguridad que garanticen la integridad de la información, a pesar de caídas del sistema o intentos de accesos no autorizados.

2.1.4 Funciones del Administrador de Bases de Datos.

- **Definir el esquema conceptual**: Es tarea del administrador de datos decidir con exactitud cual es la información que debe mantenerse en la base de datos, es decir, identificar las entidades que interesan y la información que debe registrarse acerca de esas entidades. Este proceso por lo general se denomina diseño lógico –a veces conceptual- de bases de datos. Cuando el administrador de datos decide el contenido de la base de datos en un nivel abstracto, el DBA crea a continuación el esquema conceptual correspondiente, empleando el DDL conceptual

- **Definir el esquema interno**: El DBA debe decidir también como se representará la información en la base de datos almacenada. A este proceso suele llamársele diseño físico de la base de datos. Una vez echo esto el DBA deberá crear la definición de estructura de almacenamiento correspondiente (es decir, el esquema interno) valiéndose del DDL interno. Además, deberá definir la correspondencia pertinente entre los esquemas interno y conceptual. En la práctica, ya sea el DDL conceptual o bien el DDL interno incluirán seguramente los medios para definir dicha correspondencia, pero las dos funciones (crear el esquema, definir la correspondencia) deberán poder separarse con nitidez.

- **Vincularse con los usuarios**: El DBA debe encargarse de la comunicación con los usuarios, garantizar la disponibilidad de los datos que requieren y escribir - o ayudar a los usuarios a escribir- los esquemas externos necesarios, empleando el DDL externo aplicable. Además, será preciso definir la correspondencia entre cualquier esquema externo y el esquema conceptual. En la práctica, el DDL externo incluirá con toda probabilidad los medios para especificar dicha correspondencia, pero en este caso también el esquema y la correspondencia deberán poder separarse con claridad.

Otros aspectos de la función de enlace con los usuarios incluyen las consultas sobre diseño de aplicaciones, la ayuda en la localización y resolución de problemas, y otros servicios profesionales similares relacionados con el sistema.

- **Definir las verificaciones de seguridad e integridad**: Las verificaciones de seguridad y de integridad pueden considerarse parte del esquema conceptual. El DDL conceptual incluirá los medios para especificar dichas verificaciones.

- **Definir procedimientos de respaldo y recuperación**: En caso de que sufra daño cualquier porción de la base de datos – por causa de un error humano, digamos, o una falla en el equipo o en el sistema que lo apoya – resulta esencial poder reparar los datos implicados con un mínimo de retraso y afectando lo menos posible el resto del sistema. En teoría, por ejemplo la disponibilidad de los datos no dañados no debería verse afectada.

  El DBA debe definir y poner en práctica un plan de recuperación adecuada que incluya, por ejemplo una descarga o "vaciado" periódico de la base de datos en un medio de almacenamiento de respaldo, y procedimientos para cargar otra vez la base de datos a partir de vaciado más reciente.

- **Supervisar el desempeño y responder a cambios en los requerimientos**: Es responsabilidad del DBA organizar el sistema de modo que se obtenga el mejor desempeño, y realizar los ajustes apropiados cuando cambien los requerimientos.

2.1.5 Esquema Lógico y Físico de base de datos.

Nivel físico.

Es la representación del nivel más bajo de abstracción, en éste se describe en detalle la forma en como se almacenan los datos en los dispositivos de

almacenamiento (por ejemplo, mediante índices para el acceso aleatorio a los datos).

Nivel conceptual.

El siguiente nivel más alto de abstracción, describe que datos son almacenados realmente en la base de datos y las relaciones que existen entre los mismos, describe la base de datos completa desde el punto de vista de su estructura de diseño. El nivel conceptual de abstracción lo usan los administradores de bases de datos, quienes deben decidir qué información se va a guardar en la base de datos.

Consta de las siguientes definiciones:

- **Definición de los datos**: Se describen el tipo de datos y la longitud de campo todos los elementos direccionables en la base. Los elementos por definir incluyen artículos elementales (atributos), totales de datos y registros conceptuales (entidades).

- **Relaciones entre datos**: Se definen las relaciones entre datos para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples.

En el nivel conceptual la base de datos aparece como una colección de registros lógicos, sin descriptores de almacenamiento. En realidad los archivos conceptuales no existen físicamente. La transformación de registros conceptuales a registros físicos para el almacenamiento se lleva a cabo por el sistema y es transparente al usuario.

Nivel de visión.

Nivel más alto de abstracción, es lo que el usuario final puede visualizar del sistema terminado, describe sólo una parte de la base de datos al usuario acreditado para verla. El sistema puede proporcionar muchas visiones para la misma base de datos.

Ventajas de un sistema de base de datos son:

Control de la redundancia e inconsistencia de datos.

Puesto que los archivos que mantienen almacenada la información son creados por diferentes tipos de programas de aplicación existe la posibilidad de que si no se controla detalladamente el almacenamiento, se pueda originar un duplicado de información, es decir, que la misma información sea más de una vez en un dispositivo de almacenamiento. Puede originar la inconsistencia de los datos esto es diversas copias de un mismo dato no concuerdan entre si.

Facilidad de acceso a los datos.

Un sistema de base de datos debe contemplar un entorno de datos que le facilite al usuario el manejo de los mismos.

Acceso concurrente de datos.

Para mejorar el funcionamiento global del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que múltiples usuarios actualicen los datos simultáneamente. Para prevenir esta posibilidad debe mantenerse alguna forma de supervisión en el sistema.

Seguridad de la información.

La información de toda empresa es importante, aunque unos datos lo son más que otros, por tal motivo se debe considerar el control de acceso a los mismos, no todos los usuarios pueden visualizar alguna información, por tal motivo para que un sistema de base de datos sea confiable debe mantener un grado de seguridad que garantice la autentificación y protección de los datos.

Mantenimiento de integridad.

Los valores de datos almacenados en la base de datos deben satisfacer cierto tipo de restricciones de consistencia. Estas restricciones se hacen cumplir en el sistema añadiendo códigos apropiados en los diversos programas de aplicación.

Existen diferentes niveles de abstracción para simplificar la interacción de los usuarios con el sistema; Interno, conceptual y externo, específicamente el de almacenamiento físico, el del usuario y el del programador.

2.2 Sistema Manejador de Base de Datos. (DBMS).

El sistema manejador de bases de datos (DBMS, Database Management System). Es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos.

El objetivo primordial de un sistema manejador base de datos es proporcionar un contorno que sea a la vez conveniente y eficiente para ser utilizado al extraer, almacenar y manipular información de la base de datos.

Todas las peticiones de acceso a la base, se manejan centralizadamente por medio del DBMS, por lo que este paquete funciona como interfase entre los usuarios y la base de datos.

2.2.1 Componentes de RDBMS.

Se compone de un DDL (lenguaje de definición de datos), de un DML (lenguaje de manipulación de datos) y de un DCL (lenguaje de control de datos), un DD (diccionario de datos).

DDL: Es una sentencia SQL la cual tiene por finalidad la definición de las estructuras para el almacenamiento de datos. Puede permitir la creación, eliminación de las estructuras de los distintos objetos de la base de datos, como tablas, vistas, espacios de tabla, etc.

Ejemplo: *create table* ***empleado***

DML: Juntamente con DDL forman el lenguaje SQL. DML es el subconjunto dedicado a la manipulación de los datos almacenados en las estructuras

definidas con DDL. El DML se usa para acceder a los datos (consultar, insertar, modificar y borrar).

Ejemplo:

insert table ***empleado***

DCL: Es una sentencia de SQL, que permite controlar los accesos a la base de datos.

Ejemplo:

grant

DD: En un programa de administración de bases de datos, lista de todos los archivos de bases de datos, índices, vistas y otros archivos importantes para una aplicación de base de datos.

Ejemplo:

Todo objeto del sistema y objeto creado dentro de las bases: tablas del sistema, tablas de usuario, vistas, constraints, procedimientos, etc.

2.3 Modelos de Datos.

Un modelo es una representación de la realidad que contiene las características generales de algo que se va a realizar.

Un Modelo de Datos es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos.

Los sistemas administradores de bases de datos convencionales usan uno de los tres modelos lógicos de bases de datos para hacer seguimiento de las entidades, atributos y relaciones.

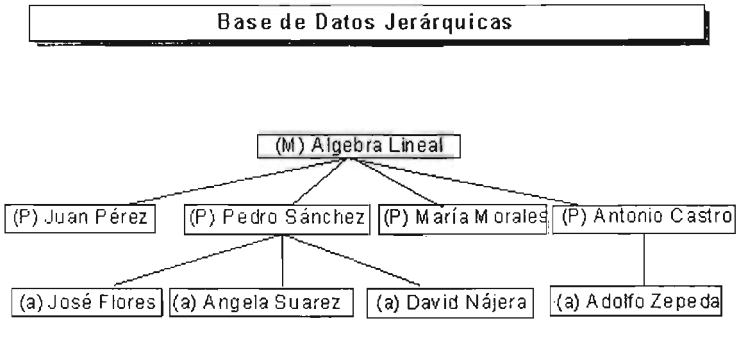
2.3.1 Modelos lógicos basados en registros.

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

Los tres modelos mas ampliamente aceptados son los siguientes:

2.3.2 Modelo jerárquico de datos.

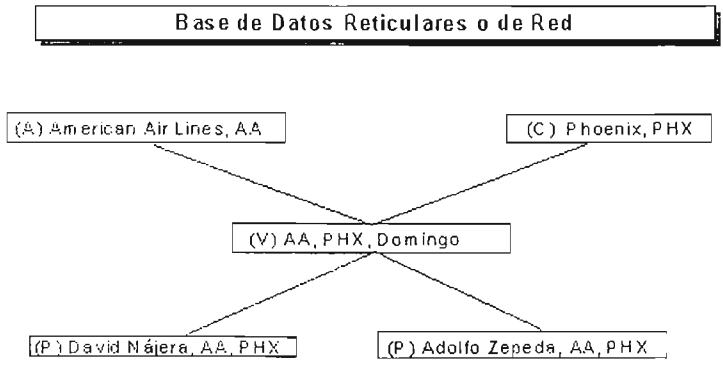
Una clase de modelo lógico de bases de datos que tiene una estructura arborescente. Un registro subdivide en segmentos que se interconectan en relaciones padre e hijo y muchos más. Los primeros sistemas administradores de bases de datos eran jerárquicos. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos.



Modelo de datos Jerárquico.

2.3.3 Modelo de datos en red.

Es una variación del modelo de datos jerárquico. De hecho las bases de datos pueden traducirse de jerárquicas a en redes y viceversa con el objeto de optimizar la velocidad y la conveniencia del procesamiento. El Modelo representa los datos mediante colecciones de registros y sus relaciones se representan por medio de ligas o enlaces.

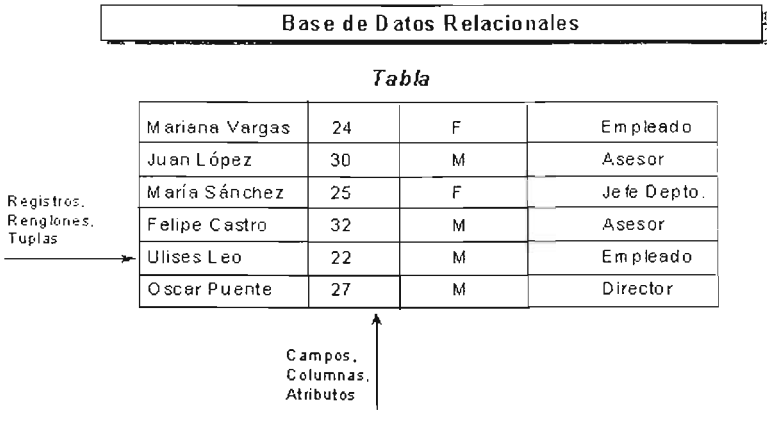


Modelo de datos en Red.

2.3.4 Modelo relacional de datos.

Es él más reciente de estos modelos, supera algunas de las limitaciones de los otros dos anteriores. El modelo relacional de datos representa todos los datos en la base de datos como sencillas tablas de dos dimensiones llamadas relaciones.

Las tablas son semejantes a los archivos planos, pero la información en más de un archivo puede ser fácilmente extraída y combinada.



Modelo Relacional de Datos.

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

Con la irrupción masiva en el mercado de los micro-ordenadores, aparecieron algunas implementaciones de RDBMS que intentaban emular las propiedades de los grandes sistemas, aunque no contaban con la mayor parte de las características necesarias para ser denominados "relacionales", especialmente en lo que se refiere al cumplimiento de las reglas de integridad relacional.

2.3.5 Modelos lógicos basados en objetos.

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

2.3.6 Modelo Entidad – Relación.

El modelo Entidad-Relación incorpora información conceptual esencial acerca del mundo real. Una vista natural del mundo real se compone de entidades y relaciones. Este modelo representa a la realidad a través de entidades, que son objetos que existen y que se distinguen de otros por sus características. Los siguientes son conceptos básicos del modelo de Entidad-Relación:

- Una entidad es algo que puede ser claramente identificado con respecto a la información que se necesita tener, puede representar algo real, tangible o abstracto.

- Una relación es una asociación entre entidades. Dos entidades conectadas por un verbo o una preposición usualmente implica una relación.

- Las entidades y las relaciones tienen diversos atributos. Estos atributos están derivados de la información identificada a través del Análisis de Sistemas.

Las entidades pueden ser de dos tipos:

Tangibles: Son todos aquellos objetos físicos que podemos ver, tocar o sentir.

Intangibles: Todos aquellos eventos u objetos conceptuales que no podemos ver, aun sabiendo que existen.

Las características de las entidades en base de datos se llaman atributos. Una entidad se caracteriza y distingue de otra por los atributos, en ocasiones llamadas propiedades, que representan las características de una entidad. Así cada entidad se describe por medio de un conjunto de parejas formadas por el atributo y el valor de dato. Habrá una pareja para cada atributo del conjunto de entidades.

Una relación es la asociación que existe entre dos a más entidades.

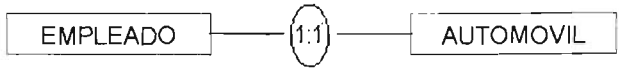
2.4 Tipos de asociaciones.

Las entidades pueden asociarse mediante relaciones. El modelo E – R contiene tanto relaciones como instancias relaciónales.
Las clases de relaciones son asociaciones entre las clases de entidad, y las instancias de relaciones son asociaciones entre las instancias de entidad. Las relaciones tienen atributos.

Una clase de relación puede involucrar muchas clases de entidades. El número de estas en la relación es el grado de estas.

Asociaciones uno a uno: Se presenta cuando existe una relación como su nombre lo indica uno a uno. Una entidad del tipo A solo se puede relacionar con una entidad del tipo B, y viceversa.

En el siguiente ejemplo la relación de ASIGNACIÓN – AUTO (AUTO-ASSIGNMENT) asocia a un único EMPLEADO con un solo AUTO. De acuerdo a este diagrama, ningún empleado tiene más de un automóvil asignado y ningún automóvil se asigna a más de un empleado.

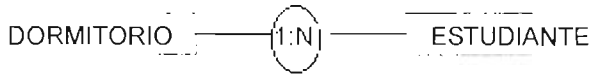


Asociación uno a uno.

Asociación uno a muchos: Significa que una entidad del tipo A puede relacionarse con cualquier cantidad de entidades del tipo B, y una entidad del tipo B solo puede estar relacionada con una entidad del tipo A.

En el siguiente ejemplo se denomina DORMITORIO – OCUPANTE, una sola instancia de DORMITORIO relaciona a muchas instancias de ESTUDIANTE. De acuerdo con el esquema, un dormitorio tiene muchos estudiantes, pero un estudiante tiene solo un dormitorio. Las posiciones de 1 y de la N son significativas.

El 1 esta cerca de la línea que conecta con DORMITORIO, lo cual significa que el 1 se refiere al lado del DORMITORIO de la relación, y la N esta cerca de la línea que conecta con ESTUDIANTE, lo cual significa que la N se refiere a la parte ESTUDIANTE de la relación.

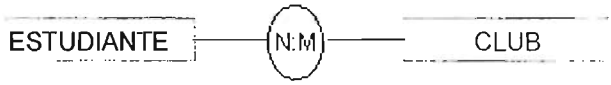


Asociación uno a muchos.

Muchos a muchos: Establece que cualquier cantidad de entidades del tipo A pueden estar relacionados con cualquier cantidad de entidades del tipo B.

En este ejemplo se muestra la relación N:M en donde se le llama al ESTUDIANTE – CLUB y la relación de instancias de ESTUDIANTE con las de

CLUB. Un estudiante puede reunirse en más de un club, y un club puede tener muchos estudiantes.



Asociación muchos a muchos.

A estos tipos de asociaciones antes descritos, también se le conoce como cardinalidad, la cual nos especifica los tipos de relaciones que existen entre las entidades en el modelo E-R y establecer con esto las validaciones necesarias para conseguir que los datos de la instancia (valor único en un momento dado de una base de datos) correspondan con la realidad.

2.5 Normalización.

La normalización es una técnica enfocada al análisis de entidades para transformarlas en entidades con características más deseables. Esta basada en un conjunto de reglas aplicadas a la asociación entre atributos y entidades.

Con la normalización se logra:

- Minimizar la inconsistencia y redundancia de los datos.
- Minimizar el impacto de futuros cambios en los datos.
- Minimizar el mantenimiento de la base de datos.
- Maximizar la estabilidad del modelo de datos.

Cada forma normal hace que los datos estén más organizados que la forma anterior, por esta razón, cada forma normal debe ser completada antes de que la subsecuente pueda ser aplicada.

Dependencia funcional.

Se dice que un atributo B depende funcionalmente de A (A -> B) si cada valor de A se corresponde con un único valor de B o, visto de otra manera, si dado A puedo obtener B. Un caso típico podría ser DNI -> Nombre, pues dado un DNI puedo obtener el nombre de la persona con ese DNI.

Algunas de estas reglas se pueden realizar entre atributos para poder obtener dependencias funcionales adicionalmente. Vamos a suponer que T es una tabla relacional y X, Y, Z son subconjuntos de atributos de la tabla T.

Reflexividad: Si los valores de un subconjunto de atributos Y están incluidos dentro de un subconjunto de atributos X, se dice que X depende funcionalmente de Y (Y -> X)

Aumentación: Si un subconjunto X depende funcionalmente de otro Y, dicha dependencia se mantendrá aunque se añada otro atributo a los dos subconjuntos (X -> Y entonces X.a -> Y.a)

Transitividad: Si Y depende funcionalmente de X y Z depende funcionalmente de Y, entonces Z depende funcionalmente de X (X -> Y e Y -> Z entonces X -> Z). Por ejemplo, DNI -> NOMBRE y NOMBRE -> DIRECCIÓN, luego DNI -> DIRECCIÓN

Dependencia funcional total: Un atributo Y tiene una dependencia funcional total con otro atributo X si tiene una dependencia funcional con X y no depende funcionalmente de ningún subconjunto de X.

Por ejemplo, una empresa tiene empleados y que una persona puede ser empleado de varias empresas. Según esto, podríamos decir que DNI.EMPRESA -> NOMBRE, pero esta dependencia no es total porque también es cierto que DNI -> NOMBRE. Sin embargo, no se puede identificar el sueldo de un empleado sin saber a qué empresa pertenece, por tanto, DNI.EMPRESA -> SUELDO sí es una dependencia funcional total.

2.5.1 Primera Formal Normal.

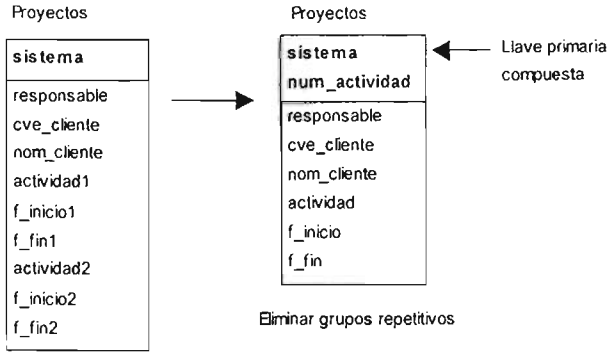
1. Identificar y separar los grupos repetitivos a otra entidad. Un grupo repetitivo es un conjunto de atributos que ocurre múltiples veces dentro de una entidad.

2. La llave primaria de esta otra entidad puede ser una llave compuesta, formada por la llave primaria de la entidad donde originalmente residía el grupo repetitivo y por la llave misma de éste.

3. La llave primaria de la nueva entidad también puede ser otra llave única, basada en las necesidades de la aplicación.

4. El nombre inicial de esta nueva entidad puede estar formado por una combinación del nombre del grupo repetitivo y del nombre de la entidad original.

Ejemplo de Conversión hacia la Primera Forma Normal



Conversión de la Primer Forma Normal.

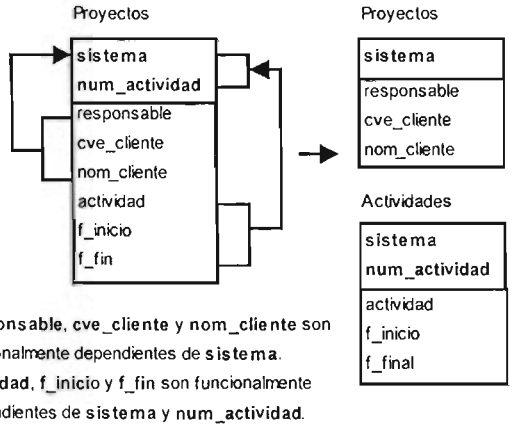
2.5.2 Segunda Formal Normal.

1. Cumplir con la primera forma normal.

2. Cada atributo es funcionalmente dependiente de la llave primaria.

Identificar y separar a otra entidad aquellos atributos que solo son parcialmente dependientes en la llave primaria o que también son dependientes de otra u otras llaves.

Ejemplo de Conversión hacia la Segunda Forma Normal



responsable, **cve_cliente** y **nom_cliente** son funcionalmente dependientes de **sistema**. **actividad**, **f_inicio** y **f_fin** son funcionalmente dependientes de **sistema** y **num_actividad**.

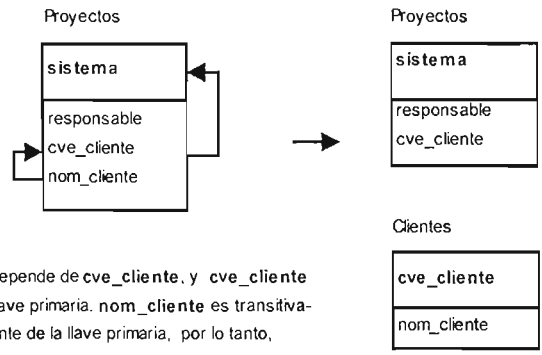
Conversión de la Segunda Forma Normal.

2.5.3 Tercera Forma Normal.

1. Cumplir con la segunda forma normal.

2. Identificar y separar en otra entidad aquellos atributos que son dependientes de una llave que no es la llave primaria, la cual es dependiente de la llave primaria.

Ejemplo de Conversión hacia la Tercera Forma Normal



nom_cliente depende de **cve_cliente**, y **cve_cliente** depende de la llave primaria. **nom_cliente** es transitivamente dependiente de la llave primaria, por lo tanto, **nom_cliente** se coloca en otra entidad.

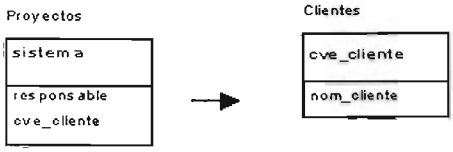
Conversión de la Tercer forma Normal.

2.5.4 Cuarta Forma Normal.

Una entidad está en Cuarta Forma Normal cuando:

1. Está en tercera forma normal y sus atributos no solamente dependen de la llave primaria o compuesta, sino que también del VALOR de la llave (dependencia del valor).

2. Un atributo cuya existencia es opcional se reubica en una entidad donde este atributo es obligatorio (dependencia de la existencia).



Cuarta Forma Normal.

2.5.5 Quinta Formal Normal.

Una entidad está en quinta forma normal si cumple con la cuarta formal normal y sus dependencias en ocurrencias de la misma entidad o tipo de entidad se han cambiado a una entidad estructurada.

2.6 Cliente/Servidor.

La arquitectura Cliente/Servidor en una base de datos, es una combinación de hardware y software cuya utilidad se reduce sino cuenta con medios de acceso a los datos.

De igual forma lo podemos entender de la siguiente manera el término cliente-servidor como un sistema en el que una máquina cliente solicita a una segunda máquina llamada servidor que ejecute una tarea específica, el cliente suele ser una computadora personal común, conectada a una red LAN (Local Área Network) y el servidor es, por lo general, una máquina anfitriona, como un servidor de archivos PC o un servidor de archivos de UNIX.

Por lo tanto el verdadero poder de los sistemas cliente /servidor radica en la gran variedad de aplicaciones cliente y software de desarrollo, llamados también Front –End. Estos Front–End son herramientas de desarrollo, de consultas, reporte y de análisis de datos.

Los clientes interactúan con el usuario, usualmente de forma gráfica y a su vez se comunican con los procesos auxiliares que se encargan de establecer la conexión con el servidor, de enviar el pedido, de recibir la respuesta, manejar las fallas y realizar actividades tanto de sincronización como de seguridad.

Los servidores proporcionan un servicio al cliente y devuelven los resultados, en algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente.

Una de las cosas por la que se distingue el modelo Cliente/Servidor, es que su procesamiento es distribuido entre aplicaciones independientes.

Desde luego que para realizar una comunicación entre el cliente y el servidor, debemos tener una infraestructura de comunicación que es la que nos va a proporcionar los mecanismos de direccionamiento y de transporte, la mayoría de los sistemas cliente -servidor utilizan las redes LAN.

2.7 Capas Lógicas Arquitectura Cliente/Servidor.

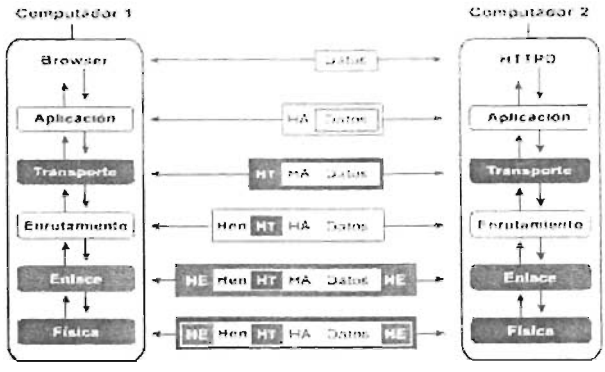
Para el caso *CLIENTE-SERVIDOR*, se utiliza un protocolo solicitud-respuesta (request/reply), en vez del OSI (TCP/IP). El cliente envía un request pidiendo un servicio y el server lo recibe, realiza el trabajo y devuelve los datos pedidos o un código de error. Lo principal es su sencillez y su eficiencia.

Sencillez: No se tiene que establecer ninguna conexión sino hasta que esta se utilice, y el mensaje de respuesta sirve como agradecimiento a la solicitud.

Eficiencia: Las *capas* del protocolo son menos y por lo tanto más eficiente, si todas las máquinas fuesen idénticas, solo se necesitarían tres niveles: La Física, La de Enlace (ambas manejadas por Hardware), la de Solicitud/Respuesta (en lugar de la de sesión.
Las capas 3 y 4 no se utilizan pues no es necesario el ruteo ni tampoco se establecen conexiones. No existe administración de la sesión puesto que no existe y tampoco se utilizan las capas superiores. Debido a que esta estructura es sencilla, se pueden reducir los servicios de comunicación que presta el micro núcleo a dos llamadas a sistema: - SEND envía el mensaje que apunta al proceso destino y bloquea al proceso llamador hasta que se envíe el mensaje; - RECEIVE que hace que se bloquee el que realizó la llamada, hasta que llegue un mensaje. El parámetro addr, determina la dirección a la cual escucha el receptor.

Podemos observar que el envío y recepción de datos entre dos aplicaciones es un proceso de intercambio de datos entre capas iguales, basado en un modelo cliente - servidor. La aplicación de la computadora 1 es una aplicación cliente "Browser / Navegador" que hace uso del protocolo "HTTP" de la capa de aplicación y la aplicación de la computadora 2 es una aplicación de servicios de páginas web que hace uso del demonio HTTPD.

Cuando la aplicación cliente requiere de los servicios de la aplicación servidor "HTTPD", dicha aplicación solicita los servicios de red al sistema operativo a través de una interfaz de programación de servicios TCP-IP.



Capas lógicas del Modelo Cliente / servidor.

Las aplicaciones cliente servidor dependen de los servicios de las capas de aplicación, transporte, enrutamiento y enlace. Cuando una aplicación hace uso de los servicios de la capa de aplicación para el envío de datos, cada capa debe de incluir: los parámetros del protocolo de comunicación entre capas adyacentes y los parámetros del protocolo de comunicación entre capas iguales. Una vez que cada capa haya incluido los parámetros antes mencionados, los datos de la aplicación estarán encapsulados. Y cuando los datos encapsulados llegan al computador destino un proceso de encapsulamiento ocurre en cada capa estableciendo de esta manera la comunicación virtual entre capas iguales.

2.7.1 Arquitectura cliente-servidor.

Un modelo de computación mediante el que las aplicaciones cliente que se ejecutan en un escritorio o en un equipo personal tienen acceso a la información contenida en servidores remotos o en equipos host. La parte cliente de la aplicación suele estar optimizada para la interacción con el usuario, mientras que la parte servidor proporciona la funcionalidad centralizada multiusuario.

2.7.2 Arquitectura de dos capas.

El modelo Cliente/Servidor corresponde a una arquitectura con dos capas: el servidor SQL por un lado y las aplicaciones que se conectan como clientes del servidor por otro.

La mayoría de las aplicaciones Cliente-Servidor funcionan bajo una arquitectura de dos capas en lenguajes de cuarta generación. Estas aplicaciones son separadas en las siguientes capas: El llamado Front-End (la interfaz del usuario, llamadas a SQL, aplicación de escritorio, etcétera) y el llamado Back-end (servidor de Bases de datos SQL, Sistema operativo multitareas, etc.).

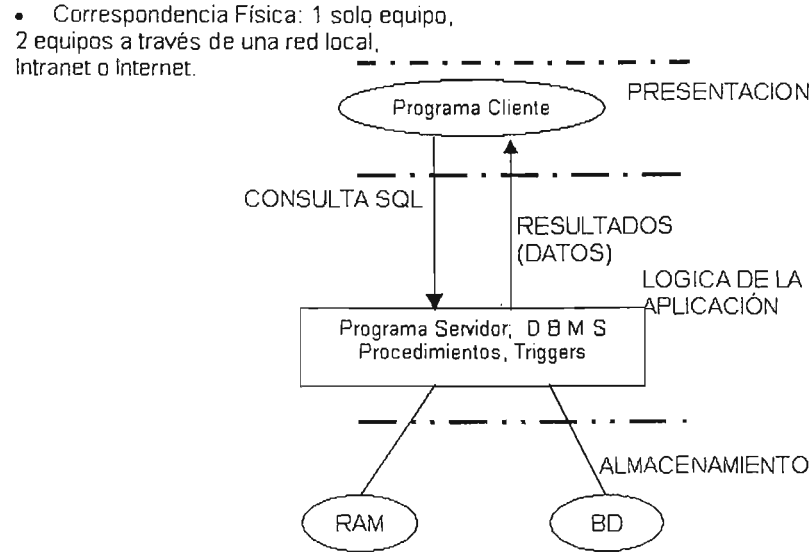
El proceso front-end se desarrolla en algún lenguaje de 4ª generación (4GL) como Visual Basic. Se llama front-end dado que es la capa en donde el usuario interactúa con su PC. El proceso back-end es el servidor de bases de datos como SQL Server ú Oracle. Se llama así dado que típicamente reside en un servidor central en un entorno controlado. Uno de los problemas es la dificultad de manipular los cambios en el front-end.

Es decir, ¿Qué pasa si desea alterar las consultas SQL dado que una columna ha sido añadida a la tabla de Base de datos? ¿Y qué cuando los más antiguos distribuidores ahora serán marcados como inactivos (no eliminados) de la base de datos?

En estos casos, varios (a veces decenas, tal vez cientos o miles) de estaciones de trabajo clientes necesitarán ser actualizadas con una nueva versión del front-end simultáneamente al cambio en la base de datos. Este no es un cambio sencillo, sobre todo si las aplicaciones cliente están geográficamente dispersas. Otro problema es la dificultad de compartir procesos comunes.

La seguridad. Esta puede ser establecida por el back-end del servidor de bases de datos o por la aplicación que sirve de front-end. Cada una tiene sus limitaciones: El primero consiste en dar privilegios a los objetos de la base de

datos y a los usuarios. Sin embargo, las corporaciones no requieren sólo asegurar cuales datos pueden ser actualizados o accedidos, sino cómo. En cuanto al segundo punto, que es el más usado, aunque el usuario puede acceder a la base de datos con su identificación, tiene dos problemas: 1. Dado que ninguno de los objetos en la base de datos es seguro, cualquier usuario pudiese tener acceso total a la misma con alguna otra herramienta de front-end (Como Excel, Access, etcétera.); 2. La implantación de la seguridad deberá ser desarrollada, probada y mantenida en absolutamente toda la red (no importa dónde se encuentren las estaciones cliente).



Arquitectura de dos capas.

2.7.3 Arquitectura de tres capas.

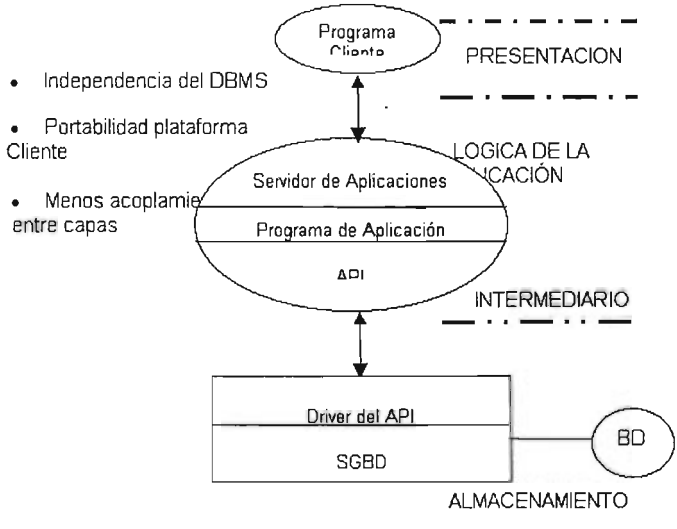
Divide una aplicación de red en tres áreas lógicas. La primera capa se encarga de la presentación, la segunda procesa la lógica de negocios y la tercera procesa los datos.

Una arquitectura de 3 capas (o de N capas) se define también como el modelo de servicios. Verá, las bases de datos, las herramientas de desarrollo y los corporativos se están moviendo hacia esta arquitectura dadas las limitaciones de la de dos capas.

La capa adicional provee de una capa explícita para las reglas de los negocios que se sitúa entre lo que se ha llamado front-end y back-end. Esta capa intermedia encapsula el modelo de negocios asociado con el sistema y le separa de la presentación y el código de bases de datos.

En la documentación de Visual Basic, Microsoft llama adecuadamente a las aplicaciones cliente "Servicios del usuario" dado que son estas las que interactúan directamente con los usuarios finales.

La arquitectura de 3 capas depende en gran parte de los procesos de comunicación. Un común denominador debe existir para permitir cualquier proceso. La comunicación deberá ser local (en las mismas máquinas) o remota (en máquinas distantes). Deberá ser transparente al proceso que envuelve sin importar si es local o remota.



Arquitectura de Tres capas.

2.7.4 Arquitectura multicapa.

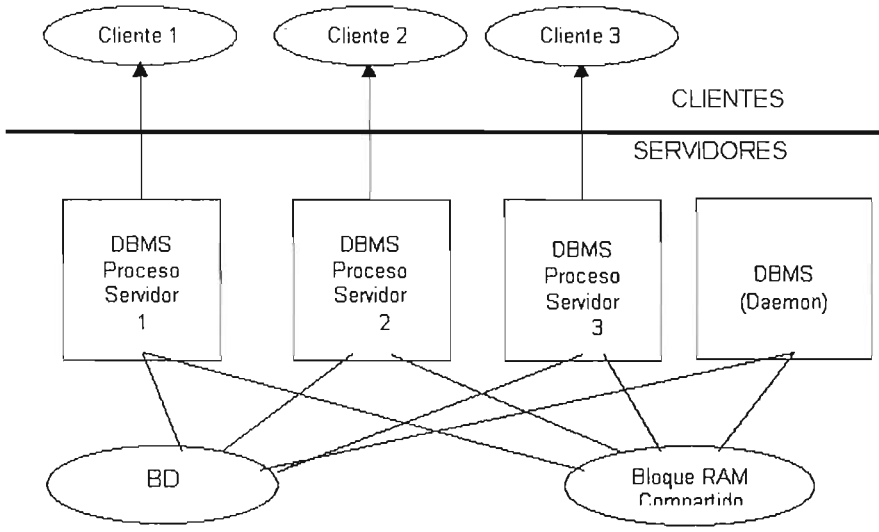
Conocida también como arquitectura de tres capas, la arquitectura multicapa es una técnica para crear aplicaciones generalmente divididas en capas de servicios de usuario, de negocios y de datos.

La primera capa es la de presentación y esta compuesta por los componentes Web instalados en el servidor de aplicaciones. Dichos componentes son las páginas HTML.

Las páginas de HTML se manejan del lado del servidor generando el HTML que viaja al cliente, permitiendo así que en el cliente sólo haya HTML.

En la siguiente capa, donde corre la lógica del negocio, asegurando integridad transaccional. Esta capa funciona básicamente como un framework para manejo de objetos. Y finalmente la tercera capa es la compuesta por las Bases de Datos.

En un entorno Cliente/ Servidor, un ejemplo de la clase de información enviada, desde el nivel de Red en el ordenador A al nivel de Red del ordenador B, podría ser una dirección de red y quizás alguna información del chequeo de errores añadida al paquete. La interacción entre niveles adyacentes ocurre a través de una interfaz. La interfaz define que servicios ofrece el nivel más bajo al más alto y cómo serán accedidos esos servicios. Es más, cada nivel en un ordenador actúa como si se comunicara directamente con el mismo nivel en otro ordenador.



Arquitectura Multiusuario.

2.8 Capas del Nivel OSI.

Nivel de Aplicación.

El nivel más alto del modelo OSI, es el nivel de Aplicación. (Donde se ejecuta la aplicación) Sirve como ventana para que los procesos de aplicación accedan a los servicios de red. Esta capa representa los servicios que directamente soportan las aplicaciones de usuario, como software para transferencia de ficheros, para acceso a bases de datos y para e-mail. Los niveles más bajos soportan esas tareas interpretadas en el nivel de aplicación. El nivel de Aplicación maneja los accesos generales a la red, control de flujo y recuperación de errores.

Nivel de Presentación.

Determina el formato usado para intercambiar datos entre ordenadores de red. Puede ser llamado el nivel traductor (Conversación). En el ordenador que envía, este nivel traduce datos desde un formato enviado hacia abajo por el nivel de Aplicación dentro de un comúnmente reconocido, formato intermediario. En el ordenador que recibe, este nivel traduce el formato intermediario en un formato

útil para el nivel de Aplicación de ese ordenador. El nivel de Presentación es responsable de la conversión de protocolo, traducción de los datos, de encriptar los datos, cambiar o convertir el set de caracteres y expandir los comandos gráficos. El nivel de Presentación también maneja compresión de datos para reducir el número de bits que necesitan ser transmitidos.

Nivel de Sesión.

Permite a dos aplicaciones en diferentes ordenadores establecer, usar, y finalizar una conexión llamada sesión. Este nivel desempeña reconocimiento de nombre y las funciones, como seguridad, necesarias para permitir a dos aplicaciones comunicarse a través de la red.

El nivel de Sesión provee sincronización entre tareas de usuario situando checkpoints en la corriente de datos. De esta forma, si la red falla, sólo el dato después del último checkpoint tiene que ser retransmitido. Este nivel también implementa diálogo de control entre procesos de comunicación, regulando qué lado transmite, cuando y cuán largo, y así sucesivamente.

Nivel de Transporte.

Proporciona un nivel de conexión adicional, inferior al del nivel de Sesión. El nivel de Transporte asegura que los paquetes son transmitidos libres de error, en secuencia y sin pérdidas o duplicaciones. Este nivel reempaqueta mensajes, dividiendo mensajes largos en varios paquetes y juntando pequeños paquetes juntos en otro paquete.

El nivel de Transporte proporciona control de flujo, manejo de errores y está involucrado/envuelto en solventar los problemas concernientes a la transmisión y recepción de paquetes.

Nivel de Red.

Responsable de direccionar mensajes y traducir direcciones lógicas y nombres en direcciones físicas. Este nivel también determina la ruta desde el origen al ordenador destino. Determina que camino deberían tomar los datos basado en

las condiciones de la red, prioridad del servicio y otros factores. También maneja problemas de tráfico en la red, como packet switching, enrutamiento y control de congestión de datos.

Si la tarjeta de red en el router no puede transmitir un pedazo (un trozo) de datos tan largo como lo envía el ordenador, el nivel de Red en el router lo compensa rompiendo el dato en pequeñas unidades. En el ordenador destino, el nivel de Red reensambla los datos.

Nivel de Enlace.

Envía tramas de datos desde el nivel de Red al nivel Físico (Comunicación entre dos maquinas). En el lado receptor, empaqueta los bits en crudo desde el nivel Físico en tramas de datos. Una trama de datos es una estructura organizada y lógica en la que los datos pueden ser situados.

En una trama de datos simple, el ID del emisor representa la dirección del ordenador que está enviando la información; el ID del destinatario representa la dirección del ordenador al que se está enviando la información. La información de control es usada para el tipo de trama, enrutamiento e información de segmentación.
El chequeo de redundancia cíclica CRC representa corrección de errores e información de verificación para asegurar que la trama de datos es recibida propiamente.

El nivel de Enlace es responsable de proporcionar transferencia libre de errores de las tramas desde un ordenador a otro a través del nivel Físico. Esto permite al nivel de Red asumir virtualmente transmisiones libres de errores sobre la conexión de red.

Generalmente, cuando el nivel de Enlace envía una trama, espera un acuse de recibo desde el receptor. El nivel de Enlace del receptor detecta cualquier problema con la trama que pueda haber ocurrido durante la transmisión.

Las tramas que no obtuvieron acuse de recibo, o las que fueron dañadas durante la transmisión, son reenviadas.

Nivel Físico.

Nivel 1, el nivel más bajo del modelo OSI, es el nivel Físico. Este nivel transmite la corriente no estructurada de bits en crudo sobre un medio fisico (como es el cable de red). El nivel Físico relaciona las interfaces eléctricas, ópticas, mecánicas y funcionales con el cable. El nivel Físico también transporta las señales que transmiten datos, generadas por todos los niveles más altos.

Este nivel define cómo está conectado el cable a la tarjeta de red. Por ejemplo, define cuantos pines tiene el conector y la función de cada pin. También define que técnica de transmisión será usada para enviar datos por el cable de red.

El nivel Físico es responsable de transmitir bits (ceros y unos) de un ordenador a otro. Los bits en sí mismos no tienen un significado definido en éste nivel. Este nivel define 'encodificación' de datos (data encoding) y sincronización de bit, asegurando que cuando un host transmisor envía un bit 1, se recibe como un bit 1, no un bit 0. Este nivel también define cuán larga es la duración de un bit y cómo cada bit es trasladado en el impulso apropiado eléctrico u óptico para el cable de red.

Arquitectura de 3 capas

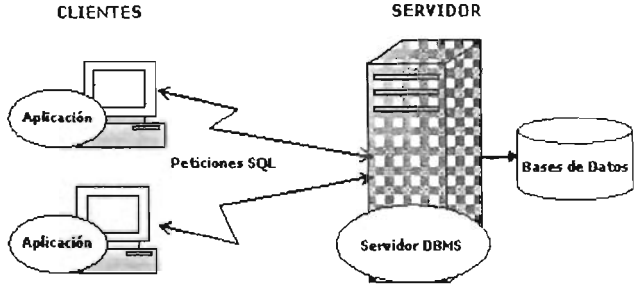
- **El cliente puede ser :**
  - Browser.
  - Applet.
  - Aplicación.
- **El servidor de Aplicaciones dispara:**
  - Procesos que se comunican con la BD.
  - Threads que se comunican a la BD.
  - Páginas Dinámicas (Paginas de Servidor).

- Componentes no distribuidos.
- Componentes distribuidos.

2.9 Componentes Cliente/Servidor en las Bases de Datos.

Como su nombre lo indica esta compuesta por Clientes y Servidores, pero, además, oculto en este nombre se encuentra los mecanismos de interacción entre ellos, clave en las aplicaciones de este tipo. Se denomina Middleware.

"*Middleware* es un término vago que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores. Como el software que ocupa la parte intermedia del sistema de cliente/servidor. Es el enlace que permite que un cliente obtenga un servicio de un servidor.



Componentes Cliente / Servidor.

El cliente envía mensajes que representados en solicitudes SQL hacia el servidor de bases de datos. Los resultados de cada orden de SQL son devueltos al cliente.

El DBMS se encarga de recolectar los datos desde su base de datos, no envía los registros completos, teniéndose un uso mucho más eficiente de la capacidad de procesamiento distribuida. Es usual que se generen aplicaciones en el cliente y en el servidor. Los servidores de bases de datos constituyen el fundamento de

los sistemas de apoyo de decisiones que precisan de consultas específicas y reportes flexibles.

2.10 SQL

El nombre "SQL" es una abreviatura de Structured Query Language (Lenguaje de consultas estructurado).

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. Como su propio nombre indica, SQL es un lenguaje informático que se puede utilizar para interaccionar con una base de datos y más concretamente con un tipo especifico llamado base de datos relacional.

El objetivo principal de SQL es la realización de consultas y cálculos con los datos de una o varias tablas en una base de datos, además de ser muy flexible y tener una sintaxis muy especifica.

El SQL ocupa dos tipos de lenguajes:

Procedural:

Con un LMD procedural el usuario (normalmente será un programador) especifica qué datos se necesitan y cómo hay que obtenerlos.

Esto quiere decir que el usuario debe especificar todas las operaciones de acceso a datos llamando a los procedimientos necesarios para obtener la información requerida. Estos lenguajes acceden a un registro, lo procesan y basándose en los resultados obtenidos, acceden a otro registro, que también deben procesar. Así se va accediendo a registros y se van procesando hasta que se obtienen los datos deseados. Las sentencias de un LMD procedural deben estar embebidas en un lenguaje de alto nivel, ya que se necesitan sus estructuras (bucles, condicionales, etc.) Para obtener y procesar cada registro individual. A este lenguaje se le denomina lenguaje anfitrión. Las bases de datos jerárquicas y de red utilizan LMD procedurales.

PL/SQL (Procedural Language /SQL) es una extensión de SQL obteniéndose como resultado un lenguaje estructural más poderoso que SQL. La unidad de programación utilizada por PL/SQL es el bloque. Todos los programas de PL/SQL están conformados por bloques. Típicamente, cada bloque lleva a cabo una acción lógica en el programa. Un bloque tendrá siempre la siguiente estructura:

```
DECLARE
//Sección declarativa: variables, tipos, y subprogramas de uso local

BEGIN
//Sección ejecutable: las instrucciones procedímentales, y de SQL aparecen aquí.
Es la única sección obligatoria en el bloque.

EXCEPTION
//Sección de manejo de excepciones. Las rutinas de manejo de errores
//aparecen aquí
END;
```

Solo se requiere que aparezca la sección ejecutable. Lo demás es opcional. Las únicas instrucciones SQL permitidas en un bloque PL/SQL son INSERT, UPDATE, DELETE y SELECT, además de algunas instrucciones para manipulación de datos, e instrucciones para control de transacciones. Otras instrucciones de SQL como DROP, CREATE o ALTER no son permitidas. Se permite el uso de comentarios estilo /* . . .*/. PL/SQL no es "case sensitive" por lo que no hay distinción entre nombres con mayúsculas y minúsculas.

En la sección de declaraciones, se indican las variables que serán usadas dentro del bloque y sus tipos. Por ejemplo:

```
DECLARE
myBeer VARCHAR (20);
price NUMBER (6,2);
```

En algunos casos, es posible que se desee que el tipo de una variable coincida con el tipo usado para una columna de una tabla determinada, en esos casos se puede usar la construcción:

DECLARE
myBeer Beers.name%TYPE;

Con lo cual se logra que la variable myBeer tenga el mismo tipo que la columna name de la tabla Beers.

También es posible inicializar las variables, mediante el operador:=. Además, mediante el uso del mismo operador es posible hacer asignaciones en el cuerpo del programa. Por ejemplo:

```
DECLARE
price NUMBER:= 300;
BEGIN
price: = price + 150;
END;
.
run
```

La ejecución de este bloque no tendrá ningún efecto, ya que no se están haciendo cambios sobre la base de datos.

No Procedural:

Un LMD no procedural se puede utilizar de manera independiente para especificar operaciones complejas sobre la base de datos de forma concisa. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde una terminal o bien embeberlas en un lenguaje de programación de alto nivel. Los LMD no procedurales permiten especificar los datos a obtener en una consulta o los datos que se deben actualizar, mediante una sola y sencilla sentencia.

El usuario o programador especifica qué datos quiere obtener sin decir cómo se debe acceder a ellos. El SGBD traduce las sentencias del LMD en uno o varios

procedimientos que manipulan los conjuntos de registros necesarios. Esto libera al usuario de tener que conocer cuál es la estructura física de los datos y qué algoritmos se deben utilizar para acceder a ellos. A los LMD no procedurales también se les denomina declarativo. Las bases de datos relacionales utilizan LMD no procedurales, como SQL (Structured Query Language) o QBE (Query-By-Example).

Los lenguajes no procedurales son más fáciles de aprender y de usar que los procedurales, y el usuario debe realizar menos trabajo, siendo el SGBD quien hace la mayor parte.

La parte de los LMD no procedurales que realiza la obtención de datos es lo que se denomina un lenguaje de consultas.

2.10.1 Estándares ANSI SQL: 89, 92, 99

SQL se ha convertido en el lenguaje de consulta relacional más popular.

En 1974 Donald Chamberlain y otros definieron el lenguaje SEQUEL (*Structured English Query Language*) en IBM Research. SQL es también un estándar oficial hoy.

En 1986, el ANSI adoptó SQL (sustancialmente el dialecto SQL de IBM) como estándar para los lenguajes relacionales y en 1987 se transformó en estándar ISO. Esta versión del estándar va con el nombre de SQL/86.

En los años siguientes, éste ha sufrido diversas revisiones que han conducido primero a la versión SQL/89 y, posteriormente, a la actual SQL/92.

Actualmente, está en marcha un proceso de revisión del lenguaje por parte de los comités ANSI e ISO, que debería terminar en la definición de lo que en este momento se conoce como SQL3. Las características principales de esta nueva encarnación de SQL son la introducción de nuevos tipos de datos más complejos.

El SQL-3 SABD intenta continuar y en la medida de lo posible reemplazar los dos debido principalmente a dos razones:

- SQL-1: Rechaza el modelo relacional de datos, desconoce su significación e importancia y no otorga una firme dirección a seguir en el tema de las bases de datos orientadas a objetos.

- SQL-2: Reconoce correctamente el modelo relacional, falla al mencionar y enfatizar los desesperados esfuerzos de continuar una comúnmente aceptada corrupción del lenguaje SQL, por lo que este debe rechazarse sin descuidar el actual legado de este lenguaje al modelo relacional (esto se refiere a los múltiples agregados de SQL para otorgar soporte a las BDOO).

- SQL-3: Comienza la denominada época relacional/objeto, en forma análoga a lo sucedido con el surgimiento de los primeros documentos acerca del modelo relacional propuesto por Codd. De hecho, las ideas se encuentran firmemente fundamentadas en la tradición relacional, que lejos de intentar reemplazarlo, se propone más bien complementar este modelo relacional tan fuertemente arraigado en la teoría y la práctica de las bases de datos.

ESTÁNDARES PARA EL SQL

Estándar SQL	Descripción
SQL 89	En 1989, ANSI definió el SQL89, basado en el anterior pero con una serie de mejoras (definición de claves primarias, integridad de los datos, etc.). Definía una característica importante es la posibilidad de utilizarse a través de dos interfaces: interactivamente o dentro de programas de aplicación.

SQL 92	SQL-92 fue diseñado para ser un estándar para los sistemas manejadores de bases de datos relacionales (RDBMS o SGBDR). Esta basado en SQL-89, cuya primera versión se conoce como SQL-86. En 1992 aparece SQL2 o SQL92, la versión hoy en dia más difundida ([ISO/IEC 1992] [ANSI 1992] [ISO/IEC 1994]). Con la aparición de la segunda versión del estándar (SQL2) en 1992, prácticamente todos los SGBDR, incluso los no relacionales, incluían soporte a SQL.
SQL 99	Comprende temas como desencadenantes o disparadores (trigger), funciones, consultas recurrentes, colecciones y SQL para objetos, incluidos los tipos de datos abstractos (ADT: abstract data types) definidos por el usuario. Un ADT se asemeja a una clase de C++; se compone de un conjunto de propiedades y métodos.

COMPARACIÓN ENTRE ESTÁNDARES DEL SQL

SQL-89	SQL-92	SQL-99
Permite expresar claves primarias, claves candidato (claves únicas) y claves externas. No soporta el concepto de clave primaria (ni otro tipo de claves).	En SQL2 ya hay definidos los operadores para la diferencia (EXCEPT) y la intersección (INTERSECT).	El disparador se activa antes o después de una modificación sobre la relación (inserción, borrado, actualización o actualización de unas columnas determinadas.
PRIMARY KEY(k1,k2,...) Claves candidato UNIQUE(u1,u2,...) Claves externas FOREIGN KEY(a1,a2, ...) REFERENCES Tabla Referenciada[(k1, k2, ...)]	Las operaciones con el esquema de la base de datos (CREATE, DROP, etc.) sólo pueden ser realizadas por el propietario del esquema. CREATE DOMAIN nombre Dominio tipo Datos CHECK predicado.	Create trigger<nombre trigger> on <tabla> for (Insert/ Update/ Delete) as <transacciones SQL> Return

Se indican los atributos que forman la clave externa y a qué relación referencia. Por defecto harán referencia a la clave primaria de esta relación, pero puede indicarse exactamente a qué atributos se hace referencia.	Define un dominio que puede ser utilizado como tipo de datos de las columnas. Además se puede definir una restricción de integridad para el dominio, mediante una cláusula.	SQL-3 tiene cuatro nuevos tipos de datos LARGE OBJECT CLOB funciona como una cadena de caracteres. BLOB tiene restricciones similares.
CREATE TABLE (name char() not null, name int Null) o también name_int UNIQUE name char(20) not null REFERENCES REAL, PRIMARY KEY (,), UNIQUE (n_), FOREIGN KEY(n_) REFERENCES X);	La definición de restricciones genéricas en SQL2 se basa en la utilización de la cláusula. CHECK CONSTRAINT nombre_restricción restricción.	Complejas combinaciones de predicados pueden ser ahora expresadas de una manera más sencilla que antes. WHERE COL1>COL2 AND COL3=COL4 OR UNIQUE(COL6) IS NOT FALSE;

2.11 Formatos Electrónicos.

Uno puedo elegir un formato que requiere de software propietario-comercial o software libre para su lectura. Por lo general el software propietario-comercial es muy bueno para elaborar y leer la publicación electrónica ya que generalmente está diseñado para cumplir justo con esos propósitos. Ejemplo de esto sería la gran mayoría del software para los libros electrónicos. Desafortunadamente estos suelen tener un precio. También significa generalmente que la persona que quiere leer la publicación electrónica tiene que contar con el software adecuado para su lectura y este suele tener un costo.

Ejemplo: actualmente existe Open eBook una iniciativa para la definición de formatos y estructura de ebooks en HTML y XML. Su propósito es crear y mantener estándares y promover el uso de los libros electrónicos. Es una asociación de compañías de software y hardware, editoriales, autores y usuarios de libros electrónicos.

2.11.1 Definición del Documento Electrónico.

La tecnología ha resuelto la edición original en papel de un documento a forma electrónica, y lo entrega directamente al suscriptor/usuario por vía electrónica quien puede acceder a la información en formato electrónico o a larga distancia. Lo que aún no se ha resuelto plenamente, y existen controversias por los diferentes enfoques y la protección a diferentes intereses es lo relativo a derecho de autor, pago de regalías y facturación de servicios por los documentos disponibles en la red Internet.

El formato electrónico permite que el documento pase directamente del editor al usuario, y es posible que el usuario con hacer *clic* en un capítulo del documento que le interese, pueda consultar artículo por artículo, palabra por palabra a través de Internet.

El texto completo electrónico empieza a aplicarse con especial interés en los libros y revistas, en lo que se ha dado por llamar "libros interactivos", que permiten integrar y hacer relaciones entre una obra original, todas sus versiones, y comentarios e interpretaciones que sobre ella se han hecho.

Esta modalidad de documento, se denomina "texto completo electrónico" y se está ofreciendo a través de Internet bajo plataformas *WWW,* utilizando un lenguaje de programación denominado *HTML* (***HyperText Markup Language***), en español corresponde a Lenguaje de Marcas de Hipertexto, es decir, el lenguaje para elaborar documentos Web(WWW).

A grandes rasgos podríamos decir que los documentos electrónicos tienen las mismas ventajas que los que se distribuyen impresos en papel, pero sin sus inconvenientes. Es decir, proporcionan información; pueden tener un diseño final tan sofisticado como las que se imprimen a color en papel. Sin embargo, como no necesitan pasar por imprentas ni por canales de distribución de correo manuales, la información llega mucho antes, y, después de recibirla, podemos manipularla con nuestro sistema de procesamiento de textos.

Los documentos electrónicos son reconocidos actualmente como un medio privilegiado de difusión de la información. Por lo cual es necesario utilizar un método apropiado, adaptado a las nuevas tecnologías, para citarlos correctamente a fin de describirlos con exactitud y de fácil recuperación.

2.12 Lenguajes de Marcas.

Un lenguaje de marca o etiquetado no es lo mismo que un lenguaje de marca generalizado. Con un lenguaje de marca se describen las reglas para el procesamiento de un texto, para describir los diferentes caracteres y sus características de impresión. La marca generalizada, por otro lado, no especifica cómo deben verse las cosas, en su lugar, provee información del sistema en el que se está trabajando, con información sobre la estructura del documento únicamente.

Un lenguaje de marcado o de anotación construye un conjunto de reglas que definen todo aquello que es parte de un documento digital pero que no pertenece al texto del mismo.

El lenguaje de marcado cumple con dos objetivos esenciales para diseñar y procesar un documento digital:

- Separa un texto en los elementos en los que se compone, como por ejemplo un párrafo, un capítulo, etc.

```
<HTML>
<HEAD>
<TITLE> Mi primer página </TITLE>
</HEAD>
<BODY>
<FONT FACE = "verdana, arial, century" size ="5">
<H> Título </H>
<EM> Un párrafo </EM>
</FONT>
</BODY>
</HTML>
```

2.12.1 SGML.

SGML (*Standard Generalized Markup Language)* o Lenguaje de Marcas Generalizado, este lenguaje data de 1986, pero que empezó a gestarse desde principios de los años 70, y que a su vez basado en el GML creado por IBM en 1969, es un lenguaje que sirve para especificar las reglas de etiquetado de documentos y no impone en si ningún conjunto de etiquetas, en especial es usado para manejo de grandes documentos, maneja diferentes tipos de ligas, separa la presentación del contenido en su semántica, permite definir esquemas con etiquetas de significado arbitrario, (lenguajes de uso particular) gran flexibilidad y capacidad expresiva y de reglas de interpretación compleja.

SGML es un lenguaje para la descripción de lenguajes de etiquetado, particularmente aquellos usados en el intercambio electrónico, manejo y publicación de documentos. HTML es un ejemplo de un lenguaje definido en SGML.

SGML es utilizado desde mitad de los 80 y ha permanecido bastante estable. Gran parte de su estabilidad se la debe al hecho de que el lenguaje es a la vez flexible y rico en posibilidades. Esta flexibilidad tiene, sin embargo, su coste, el nivel de complejidad que ha inhibido su uso en diversos ámbitos como la World Wide Web.

Ejemplo:

```
<HTML><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>Mi primer documento HTML</TITLE>
</HEAD>
<BODY>
<P>¡Hola Mundo!
</BODY>
</HTML>
<HEAD>
<TITLE>Ejemplo de salto automático de pantalla</TITLE>
<META HTTP-EQUIV="Refresh"
CONTENT="5;URL=http://www.lander.es/webmaestro">
</HEAD>
<BODY>
```

Ejemplo de salto automático de pantalla. Después de 5 segundos debe saltar
AUTOMATICAMENTE a la portada de WebMaestro.
<P>Si esto no ocurre, pulsa este
<A HREF="http://www.lander.es/webmaestro">enlace</A>.
</BODY>
</HTML>

2.12.2 HTML.

HTML: Hypertext Markup Language. Lenguaje de Marca de Hipertexto. Lenguaje etiquetado (estructura y formato definido a través de etiquetas).

El HTML es un lenguaje que permite definir la forma en que un documento será visualizado en la WWW. Un archivo HTML es un documento ASCII en donde están definidas cada una de las partes que componen la página. Se establecen marcas que definen bloques que tendrán un cierto aspecto. Estas órdenes son interpretadas por un navegador que visualiza el resultado del HTML en el computador del usuario.

<HTML>: Es la etiqueta que define el inicio del documento html, le indica al navegador que todo lo que viene a continuación debe tratarlo como una serie de códigos html.

<HEAD>: Define la cabecera del documento html, esta cabecera suele contener información sobre el documento que no se muestra directamente en el navegador.

<**BODY**>: Define el contenido principal o cuerpo del documento, esta es la parte del documento html que se muestra en el navegador, dentro de esta etiqueta pueden definirse propiedades comunes a toda la página, como color de fondo y márgenes.

<**TITLE**> Dentro del encabezamiento hay información del documento, que no se ve en la pantalla principal, principalmente el título del documento. El título debe ser breve y descriptivo de su contenido, pues será lo que vean los demás cuando añadan nuestra página a su bookmark (o agenda de direcciones).

Si queremos obtener múltiples líneas en blanco no basta con repetir la etiqueta <P>, sino que hay que combinarla con la etiqueta
.

Ejemplo:

```
<HTML>
<HEAD>
<TITLE> Título de la página </TITLE>
</HEAD>
<BODY>
[Aquí van las etiquetas que visualizan la página]
</BODY>
<BR><P>
</HTML>
```

2.12.3 XML.

XML significa Lenguaje de Marcas Generalizado (Extensible Markup Language). Es un lenguaje usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre, creado por el Consorcio World Wide Web, W3C (los creadores de la www), en colaboración con un panel que incluye representantes de las principales compañías productoras de software.

El XML fue propuesto en 1996, y la primera especificación apareció en 1998. Desde entonces su uso ha tenido un crecimiento acelerado, que se espera que continúe durante los próximos años; hoy en día parece que de repente todo el mundo está usando, o quiere usar, XML.

Podemos establecer un cuadro-resumen con algunas de las diferencias significativas con respecto a los otros lenguajes que hemos mencionado.

Ejemplo:

```
<?xml version="1.0"?>
<libro>
<titulo> Cien años de soledad </titulo>
<disponible tiempo="24" unidad="horas"/>
```

```
<autor> Gabriel García Márquez </autor>
<formato> Rústica </formato>
<publicacion>1967 </publicacion>
<precio cantidad="9.99" moneda="euro"/>
<descuento cantidad="5"/>
<enlacelibro href="/exec/ISBN/84-473-0619-4"/>
</libro>

Su DTD correspondiente:
<!ELEMENT autor ( #PCDATA ) >
<!ELEMENT descuento EMPTY >
<!ATTLIST descuento cantidad CDATA #REQUIRED >
<!ELEMENT disponible EMPTY >
<!ATTLIST disponible tiempo CDATA #REQUIRED >
<!ATTLIST disponible unidad CDATA #REQUIRED >
<!ELEMENT enlacelibro EMPTY >
<!ATTLIST enlacelibro href CDATA #REQUIRED >
<!ELEMENT formato ( #PCDATA ) >
<!ELEMENT libro ( titulo | disponible | autor | formato | publicacion |
precio | descuento | enlacelibro )* >
<!ELEMENT precio EMPTY >
<!ATTLIST precio cantidad CDATA #REQUIRED >
<!ATTLIST precio moneda CDATA #REQUIRED >
<!ELEMENT publicacion ( #PCDATA ) >
<!ELEMENT titulo ( #PCDATA ) >
```

2.13 DTD.

Los DTD (**Document Type Definition**) son los documentos de definición de tipo. Estos documentos son aquellos que definen los elementos, atributos y entidades que pueden aparecer dentro del documento XML. Además, definen las reglas y limitaciones a las que deben de estar sujetos dichos elementos, atributos entidades.

Como antes se comentó, los documentos XML pueden ser válidos o bien formados, o no serlo, pero entonces no serían documentos XML. En cuanto a los válidos, ya sabemos que su gramática está definida en los DTD.

Pues bien, los DTD no son más que definiciones de los elementos que puede incluir un documento XML, de la forma en que deben hacerlo (qué elementos van dentro de otros) y los atributos que se les puede dar.

Inicialmente el uso de las DTD se debe a SGML, en el cual describían no sólo el vocabulario necesario para identificar todos los elementos de que iba a constar nuestro documento, si no que también expresaba la estructura que dichos

elementos. Con la creación de XML, que recordemos que no es más que un subconjunto de SGML, fue necesario mantener la posibilidad de describir los elementos necesarios, al igual que en SGML, por ello se importaron las DTD y todo su comportamiento.

Posteriormente, y debido al uso principalmente, se vio la necesidad de emplear otros métodos para describir esas necesidades inherentes a XML, con esta idea se creó XML Esquema, se pretendía mejorar y ampliar la utilidad de las DTD.
El DTD puede estar definido de dos formas:

1. Dentro del documento XML.

En este caso el DTD irá dentro del prologo del documento. Para definirlo lo haremos de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nombreDTD [
--definición del DTD-- ]>
```

2. En un documento externo.

Al ser un documento externo, puede ser compartido por múltiples personas. Para definirlo de esta forma deberemos de poner:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nombreDTD SYSTEM "fichero.dtd">
```

Podemos tener una mezcla de los dos. De tal manera que tengamos parte de la declaración en el propio documento y el resto en un DTD externo.

Hay que recordar que los documentos XML, que además de estar bien formados, se ajustan a un DTD, son documentos válidos. No existe el concepto "invalido".

2.14 Esquemas.

Al igual que las **DTDs**, los esquemas (schemas) describen la estructura de la información. El motivo de la creación de este nuevo estándar para realizar la labor de las DTDs es, básicamente, la utilidad. Durante un tiempo, y a falta de

otra solución más ajustada, se emplearon los mecanismos que proporcionaba SGML para modelar la información en XML.

Pero el descubrimiento de nuevas aplicaciones de XML al margen de la estructuración de documentos forzó la creación de otras soluciones que ayudasen a solventar los nuevos problemas a los que se enfrentaba el mercado.

Un *esquema* nos permite definir el tipo del contenido de un elemento o de un atributo, y especificar si debe ser un número entero, o una cadena de texto, o una fecha, etc.

Un ejemplo de un documento XML, y su *esquema* correspondiente:

```
<documento xmlns="x-schema: personaSchema.xml">
<persona id="fulano">
<nombre>Fulano Menganez</nombre>
</persona>
</documento>
```

Como podemos ver en el documento XML anterior, se hace referencia a un espacio de nombres (*namespace*) llamado "x-schema: personaSchema.xml". Es decir, le estamos diciendo al analizador sintáctico XML (*parser*) que valide el documento contra el *esquema* "personaSchema.xml".

El *esquema* sería algo parecido a esto:

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
 xmlns:dt="urn:schemas-microsoft-com:datatypes">
 <AttributeType name='id' dt:type='string' required='yes'/>
 <ElementType name='nombre' content='textOnly'/>
 <ElementType name='persona' content='mixed'>
 <attribute type='id'/>
 <element type='nombre'/>
 </ElementType>
 <ElementType name='documento' content='eltOnly'>
 <element type='persona'/>
 </ElementType>
</Schema>
```

El primer elemento del esquema define dos espacios de nombre. El primero "*xml-data*" le dice al analizador que esto es un esquema y no otro documento XML

cualquiera. El segundo "*datatypes*" nos permite definir el tipo de elementos y atributos utilizando el prefijo "*dt*".

ElementType

Define el tipo y contenido de un elemento, incluyendo los sub-elementos que pueda contener.

AttributeType

Asigna un tipo y condiciones a un atributo.

Attribute

Declara que un atributo previamente definido por **AttributeType** puede aparecer como atributo de un elemento determinado.

Element

Declara que un elemento previamente definido por **ElementType** puede aparecer como contenido de otro elemento.

Tal como hemos visto, es necesario empezar el esquema definiendo los elementos más profundamente anidados dentro de la estructura jerárquica de elementos del documento XML. Es decir, tenemos que trabajar "de dentro hacia fuera".

Visto de otra manera, las declaraciones de tipo **ElementType** y **AttributeType** deben preceder a las declaraciones de contenido **element** y **attribute** correspondientes.

2.15 Hojas de estilo para XML.

Aunque se ha establecido un modo para que podamos usar hojas de estilo CSS (Cascade Style Sheets) dentro de documentos XML, es lógico pensar que para aprovechar las características del nuevo lenguaje hace falta tener un estándar paralelo y similar asociado a él. De este modo en el W3C están trabajando sobre la nueva recomendación XSL: Extensible Style-sheet Language.

De todos modos, está bastante avanzado, de hecho empresas como Microsoft e IBM ya ofrecen soporte a algunas de sus características.

Según el W3C, XSL es "un lenguaje para transformar documentos XML", así como un vocabulario XML para especificar semántica de formateo de documentos.

En definitiva, además del aspecto que ya incluía CSS referente a la presentación y estilo de los elementos del documento, añade una pequeña sintaxis de lenguaje de script para poder procesar los documentos XML de forma más cómoda.

Al igual que con HTML hasta ahora, se pueden especificar las hojas de estilo, sean CSS o XSL, dentro del propio documento XML o reverenciándolas de forma externa.

2.16 CSSL.

CSSL significa Cascading Style-sheet Language, y es el lenguaje para páginas de estilo desarrollado por el consorcio W3 para ser usado en páginas HTML. Un documento XML puede también hacer uso de una página de estilo CSS, en forma semejante a como se hace en una página HTML.

2.17 XSL.

El lenguaje de páginas de estilo que ha sido desarrollado por el consorcio W3C, para dar formato a los documentos XML, se llama XSL, que es el acrónimo de Extensible Style-sheet Language. Una página de estilo XSL permite modificar un documento XML, produciendo un resultado que puede estar en uno de varios formatos diferentes incluyendo el propio XML, HTML o un fichero de texto (este fichero de texto puede ser código LaTeX o PostScript por ejemplo).

2.18 XHTML.

XHTML es una familia de módulos y tipos de documentos que reproduce, engloba y extiende HTML. Los tipos de documentos de la familia XHTML están basados

en XML, y diseñados fundamentalmente para trabajar en conjunto con agentes de usuario basados en XML.

Es una reformulación de las tres definiciones de tipo de documento HTML, como aplicaciones de XML. Su finalidad es ser usado como lenguaje de contenidos que es a la vez conforme a XML y, si se siguen algunas sencillas directrices, funciona en agentes de usuario conformes con HTML.

- Los documentos XHTML son conformes a XML. Como tales, son fácilmente visualizados, editados y validados con herramientas XML estándar.

- Los documentos XHTML pueden escribirse para que funcionen igual o mejor que lo hacían antes tanto en los agentes de usuarios conformes a HTML como en los nuevos agentes conformes a XHTML.

- Los documentos XHTML pueden usar aplicaciones (scripts y applets) que se basen ya sea en el Modelo del Objeto Documento de HTML o XML.

- A medida que la familia XHTML evolucione, los documentos estarán más preparados para interactuar dentro de y entre distintos entornos XHTML.

Comparativa de los Lenguajes de Marca

	HTML	SGML	XML
Gramática	Fija y no ampliable	Extensible	Extensible
Estructura	Monolítica	Jerárquica	Jerárquica
Nº de marcas	Fijas	Sin límite	Sin límite
Complejidad	Baja	Alta	Mediana
Diseño de páginas	Fijado por tags. Etiquetas con atributos CSS en DHTML	DSSSL	CSS o DSSSL
Enlaces	Simples enlaces	HyTime	Poderosos enlaces (XLL)
Exportabilidad (formatos/aplicaciones)	No	Sí	Sí
Validación	Sin validación	Obligatorio DTD	Pueden validarse

Búsquedas	Simple y a veces resuelta por *scripts* o CGI	Son posibles potentes búsquedas.	Potente búsqueda. Con capacidad para personalizarla
Indización/Catalogación de páginas Web	Sólo lo permite los atributos de la etiqueta <META>, e implementaciones como DC.	Algún proyecto como TEI, DLI, etc.	Una descripción abierta y personalizable con el RDF.

2.19 Sistemas Operativos.

Un Sistema Operativo es un programa que permite administrar los recursos de una computadora, como lo son: Memoria, CPU, dispositivos de E/S (Unidades de Discos, monitor, teclado, etc.).

Además, el Sistema Operativo es un programa que actúa como intermediario entre el usuario y el hardware de una computadora, siendo uno de sus propósitos el proporcionar un entorno en el cual el usuario pueda ejecutar programas.

En la tercera generación de computadoras nace uno de los primeros sistemas operativos con la filosofía de administrar una familia de computadoras: el OS/360 de IBM. Fue este un proyecto tan novedoso y ambicioso que enfrentó por primera vez una serie de problemas conflictivos debido a que anteriormente las computadoras eran creadas para dos propósitos en general: el comercial y el científico.

Así, al tratar de crear un solo sistema operativo para computadoras que podían dedicarse a un propósito, al otro o ambos, puso en evidencia la problemática del trabajo en equipos de análisis, diseño e implantación de sistemas grandes.

Surge también el concepto de la multiprogramación, porque debido al alto costo de las computadoras era necesario idear un esquema de trabajo que mantuviese a la unidad central de procesamiento más tiempo ocupada, así como el encolado (spooling) de trabajos para su lectura hacia los lugares libres de memoria o la

escritura de resultados. Sin embargo, los sistemas siguieron siendo básicamente sistemas de lote.

En la cuarta generación se hacen populares el MS-DOS y UNIX.

Para mediados de los 80's, comienza el auge de las redes de computadoras y la necesidad de sistemas operativos en red y sistemas operativos distribuidos. Para los 90's el paradigma de la programación orientada a objetos cobra auge, así como el manejo de objetos desde los sistemas operativos.
Las aplicaciones intentan crearse para ser ejecutadas en una plataforma específica y poder ver sus resultados en la pantalla o monitor de otra diferente (por ejemplo, ejecutar una simulación en una máquina con UNIX y ver los resultados en otra con DOS). Los niveles de interacción se van haciendo cada vez más profundos.

El principal objetivo de un Sistema Operativo es lograr que el sistema de computación se use de manera cómoda y hace que el hardware de la computadora se utilice de manera eficiente, podríamos decir que el Sistema Operativo es un programa que se ejecuta todo el tiempo en la computadora.

Historia de Windows

Año	Evento	Descripción
1985	Windows 1.0	Interfaz gráfica con menús desplegables, no había ventanas en cascada y soporte para ratón, gráficos de pantalla e impresora independientes del dispositivo, multitarea cooperativa entre las aplicaciones Windows.
1987	Windows 2.0	Fue renombrado como Windows/286, sus características fueron ventanas traslapadas y archivos PIF para aplicaciones DOS.
1987	Windows/386	Proveía la capacidad de ejecutar múltiples aplicaciones DOS simultáneamente en memoria extendida, por lo cual se caracterizaba por tener, Múltiples máquinas virtuales DOS con multitarea.

1990	Windows 3.0	Modo estándar (286), con soporte de memoria grande (large memory). Modo Mejorado 386, con memoria grande y soporte de múltiples sesiones DOS. Se agregó el Administrador de Programas y Administrador de Archivos. Soporte para Red. Soporte para más de 16 colores. Soporte para cajas de selección, menús jerárquicos y los archivos. INI privados para cada aplicación empezaron a cobrar más valor.
1990	Windows 3.1	Una versión de Windows con muchas mejoras a Windows 3.0, no hay soporte para el modo Real (8086), fuentes TrueType, capacidad para que una aplicación reinicie la máquina, soporte de API de multimedia y red.
1995	Windows 95	Provee un ambiente en el cual pueden correr aplicaciones de 32 bits.
1996	Windows CE	Es un sistema operativo nuevo, compacto y portable, construido desde las bases para posibilitar el desarrollo de un gran número de dispositivos comerciales y hogareños, dispositivos inalámbricos tales como teléfonos celulares inteligentes, y la próxima generación de consolas de video juego incluyendo reproductores de DVD, multitarea y multihilado que tiene una arquitectura abierta, otorgando un soporte a una variedad de dispositivos.
1998	Windows 98	Esta nueva versión continúa soportando 32 bits, se puede decir que la interfaz de Windows 98 es la interfaz que deja Internet Explorer 4.0 cuando se le instala en Windows 95 con la opción "Actualización de Escritorio", que es una versión mejorada de la interfaz nativa de Windows 95, soporte para Múltiples Monitores, soporte para USB.
1999	Windows NT 3.51 y 4.0	Se hizo claro que el hardware estaba cambiando muy rápido con relación al desarrollo de software. IBM y Microsoft decidieron comenzar a trabajar simultáneamente en dos productos. OS/2 versión 2 sería un refinamiento evolutivo de sistemas previos, actualizado para las nuevas características de hardware del 386. Continuaría el soporte a las aplicaciones y a los drivers de dispositivos desarrollados para el sistema previo. OS/2 versión 3 se basaría sobre Nueva Tecnología. Este sería escrito desde cero y se desarrollaría un sistema basado sobre los mejores principios de ingeniería de software. En un principio sería para CPUs Intel, pero sería portable a otros chips de CPU. Esto se convirtió en Windows NT.

2000	Windows 2000 (NT5.0)	Windows NT 5.0 se pasa a llamar Windows 2000, desde este sistema se pierde la nomenclatura Workstation y Server, tiene como características: Real soporte para Plug and Play. Servicios de Directorio. Mayor integración con Internet e Intranet.
2000	Nomenclaturas de Windows 2000	Windows 2000 Professional anteriormente NT Workstation. Windows 2000 Server anteriormente NT Server. Windows 2000 Advanced Server anteriormente NT Advanced Server. Windows 2000 Datacenter Server. Producto nuevo y que es el nuevo y más poderoso sistema operativo de Microsoft con posibilidad de hasta 16 procesadores simétricos y 64 GB de memoria física.

2.20 Sistema Operativo UNÍX

A pesar de los sistemas abiertos, la historia de UNIX está dominada por el ascenso y caída de los sistemas hardware. UNIX nació en 1969 en General Electric. A la vez, los Laboratorios Bell de AT&T había completado el desarrollo de Multics, un sistema multiusuario que falló por su gran demanda de disco y memoria. En respuesta a Multics, los ingenieros de sistemas Kenneth Thompson y Dennis Ritchie inventaron el UNIX. Inicialmente, Thompson y Ritchie diseñaron un sistema de archivos para su uso exclusivo, pero pronto lo cargaron en una Digital Equipment Corp. (DEC) PDP-7, una computadora con solo 18 Kilobytes de memoria. Este suministraba una larga serie de puertos. En 1970, fue cargado en una PDP-11, y el runoff, el predecesor del troff, se convirtió en el primer procesador de texto de UNIX. En 1971, UNIX recibió reconocimiento oficial de AT&T cuando la firma lo usó para escribir manuales.

La segunda edición de UNIX fue realizada en 1971. La segunda edición dio forma al UNIX moderno con la introducción del lenguaje de programación C y sobre los 18 meses siguientes, el concepto de los pipes. Los pipes fueron importantes por muchas razones. Representaron una nueva forma de tratamiento de datos. Desde un punto de vista moderno, los pipes son un mecanismo orientado a objetos, porque entregan datos desde un objeto, o programa, a otro objeto.

Mientras tanto, había mucha actividad con el C. El C es otro producto de los Laboratorios Bell. Fue formado a partir de conceptos de otros tres lenguajes: B, CPL (Combined Programming Lenguage) y Algol-60. A finales de 1973, después de que Ritchie añadió soporte para variables globales y estructuras, C se convirtió en el lenguaje de programación de UNIX preferente. (Brian Kernigham, quien ayudó a Ritchie a desarrollar el C, añadió la R al estándar K&R, es el estándar preferido hasta la aceptación del ANSI C).

System V

System V es la versión más ampliamente usada de UNIX. Es el descendiente directo del UNIX desarrollado por AT&T en 1969. Está actualmente en la revisión 4.1 y a menudo es referenciado como SVR4, o System V Release 4.

Historia de Unix

Año	Evento	Descripción
1965	Origen	Bell Telephone Laboratories y General Electric Company intervienen en el proyecto MAC (del MIT) para desarrollar MULTICS.
1969-71	Infancia del UNIX	El primer UNIX llamado Versión 1 o Primera edición, nace de las cenizas de MULTICS.
1972-73	Nace el C	En la Versión 2 el soporte del lenguaje C y los pipes son añadidos. En la Versión 4 el ciclo se completa con la reescritura de UNIX en C.
1974-75	El momento	Las Versiones 5 y 6 de UNIX se distribuyen a las universidades. La Versión 6 circula en algunos ambientes comerciales y gubernamentales. AT&T impone ahora pagar una licencia, a pesar de que no puede promocionar UNIX por las duras regulaciones de EEUU del monopolio telefónico de AT&T.
1977	UNIX como producto	Interactive Systems es la primera compañía comercial que ofrece UNIX.
1977	Nace BSD	1BSD incluye un Shell Pascal, dispositivos y el editor ex.
1979	Versión 7	La Versión 7 de UNIX incluye el compilador completo K&R con uniones y definiciones de tipos. Versión 7 también añade el Boume Shell.
1979	Trabajo en Red	BSD acrecentado por BBN incluye soporte para trabajar en red.
1979	Nace XENIX	Implementación para microcomputadoras ampliamente distribuido en hardware de bajo coste.

1980	Memoria Virtual	La capacidad de memoria virtual se añade en 4BSD.
1980	Nace ULTRIX	DEC realiza una versión de UNIX basado en BCD.
1980	Licencias en AT&T	La distribución de licencias abre el mercado.
1982	Atracción de los negocios	Soporte importante para procesos de transacciones desde UNIX System Development Lab.
1983	Nace System V	La versión más común de AT&T obtiene sus bases.
1984	Salida de SVR3	AT&T desata la versión más popular de System V hasta ahora.
1988	Motif vs. Open Look	Sistemas por ventanas rivales son anunciados por OSF y UI.
1988	Siguiente paso	Un UNIX gráfico usa el Kernel Mach.
1990	OSF/1 vs. SVR4	Versiones rivales de UNIX son anunciadas por OSF y UI.
1992-95	Socialización	OSF/1 abandona la escena; SVR4 se convierte en el estándar; Sun vende más estaciones de trabajo para usuarios de Motif que para usuarios de Open Windows; y crece Windows/NT de Microsoft.

2.20.1 Versiones de Unix.

Cuando surgió UNIX su utilización fue limitada a sus creadores (los Laboratorios Bell) y a las instituciones tales como universidades y escuelas superiores, donde los usuarios tuvieron suficiente pericia y motivación para mantener un sistema operativo sin soporte por parte de los creadores.

La variedad entre las versiones de UNIX es terrible. Las dos familias más importantes de versiones de UNIX son BSD y System V. BSD dio nacimiento a SunOS, quien se ha convertido ahora en el progenitor de muchas pequeñas variantes en el mercado de las SPARC. Tatung, por ejemplo, ofertó SPARC-OS, y Solbourne Computers ofertó SolOS. Con la adquisición de la división de sistemas operativos de Interactive Systems, Sun ha trasladado también SunOS a las arquitecturas Intel 386 y 486.

Versiones de Unix

Versión	Descripción
FreeBSD	Recientemente se han anunciado planes de integrar el código fuente de FreeBSD con el de BSD/OS distribuido por BSDI. Esto proporcionaría una ventaja tecnológica aún más grande al grupo de FreeBSD. Gracias a su modo de emulación de Linux, puede correr una variedad de software escrito para este último.
NetBSD	El énfasis de este grupo es la portabilidad del sistema operativo. Actualmente existen ports para casi cualquier plataforma, desde las antiguas VAX hasta las modernas iMac
OpenBSD	El énfasis de este grupo es en la seguridad, han hecho una auditoria de todo el código fuente buscando errores y fallas de seguridad. Incorporan sistemas criptográficos en su sistema operativo.
TrustedBSD	Este es un muy reciente sistema operativo, basado en FreeBSD y con la meta de alcanzar la certificación B1 del libro naranja del Departamento de Defensa de Estados Unidos.
Solaris	Es un sistema operativo que proporciona un ambiente de mayor seguridad, y un Sistema de Administración, tiene diversas características, portabilidad, compatibilidad, escalabilidad.
HP-UX	Incremento Instantáneo de la Capacidad según la Demanda, Particiones Virtuales y Particiones de Hardware para maximizar la escalabilidad. Ofrece una gestión del sistema y de la carga de trabajo sin precedentes, así como una alta interoperabilidad con entornos de desarrollo Windows y Linux. Proporcionan la capacidad de activar nuevos procesadores del sistema de forma flexible e instantánea.
AIX	Flexibilidad, sistema operativo multiplataforma. La flexibilidad para seleccionar un sistema basado en POWER contribuye a proteger la inversión en aplicaciones, procesos y habilidades. Contribuye a reducir los costes y a mejorar las soluciones de e-business al permitir la combinación de las aplicaciones Linux portables con la escalabilidad y la robustez de AIX

El énfasis de los grupos que distribuyen Linux es:

Versiones en Linux

Versión	Descripción
Caldera	El énfasis de esta distribución es la facilidad de uso e instalación para los usuarios. Se orientan más hacia el desktop, como cualquier otra distribución de Linux, puede ser usada para servidores.

Corel	Es una distribución basada en Debian, pero extensivamente modificada para hacerla tan fácil de usar como un conocido sistema operativo de Microsoft. Es quizá la distribución más fácil de utilizar para alguien que no esté familiarizado con Unix.
Debian	Es una distribución orientada más a desarrolladores y programadores. El énfasis de esta distribución es incluir en su sistema solamente software libre según la definición de la Fundación del Software Libre (FSF).
Mandrake	Es una distribución originalmente basada en RedHat que se enfoca principalmente hacia la facilidad de uso. Al igual que Corel, es recomendada para quienes no tengan mucha experiencia con sistemas Unix.
RedHat	Es la distribución más popular de Linux y para la que hay más paquetes comerciales de software. Está orientada tanto al desktop como a servidores. La mayoría de servidores de web que utilizan Linux como sistema operativo usan esta distribución.
S.u.S.e.	Es la distribución más popular en Europa y probablemente la segunda más popular del mundo. Al igual que RedHat, está orientada tanto a desktops como a servidores.
Slackware	Es una distribución de Linux que pretende parecerse a BSD desde el punto de vista del administrador de sistemas. No es una distribución muy popular ya que cuando comenzó era la más popular.
Stampede	Es una distribución enfocada al rendimiento y velocidad del sistema. No es muy fácil de usar para quién no está acostumbrado a la administración de sistemas Unix.

2.21 Lenguajes de Programación

Lenguaje de programación es un conjunto de sintaxis y reglas semánticas que definen los programas del computador.

Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un lenguaje le da la capacidad al programador de especificarle al computador, qué tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano.

Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador.

¿Filosofía de los Lenguajes?

La Filosofía del lenguaje es una rama de la filosofía y de la lingüística. No estudia el significado de las palabras en particular, o si determinadas oraciones son verdaderas o falsas. En el caso de los lenguajes de programación estudia la estructura sintáctica y la forma en que esta estructurado las líneas de código que al final nos darán como resultado una solución. Se hace preguntas sobre cuál es el significado, es decir, que significa cada una de esas sentencias y para que se ejecutan.

Para lograrlo, intenta modelar de forma lógica el lenguaje natural desde un punto de vista computacional. Dicho modelado no se centra en ninguna de las áreas de la lingüística en particular, sino que es un campo interdisciplinario, en el que participan lingüistas, informáticos especializados en inteligencia artificial, psicólogos cognoscitivos y expertos en lógica.

Cada uno de estos tipos de filosofía de los lenguajes provee modelos computacionales para una mejor aplicación en la solución de problemas reales.

2.22 GENERACIONES DE LENGUAJES.

2.22.1 Primera generación: Los primeros ordenadores se programaban directamente en código binario, que puede representarse mediante secuencias de (0 y 1) en sistema binario. Cada modelo de ordenador tiene su propio código, por esa razón se llama lenguaje de máquina.

Lenguaje de máquina: Los primeros ordenadores se programaban mediante cables o tableros de interruptores, que introducían el programa directamente en los circuitos del ordenador.

Pero este sistema era muy poco flexible y pronto se sustituyó por otros más manejables, como la cinta de papel perforado y la tarjeta de cartulina perforada. Las cintas y tarjetas se perforaban mediante máquinas provistas de teclados especiales.

El programa se escribía directamente en *código binario* y podía representarse mediante secuencias de ceros y unos (*bits*, abreviatura inglesa de binary digits o "dígitos binarios").

Como el código binario es largo y muy poco legible, los programadores en el lenguaje de máquina suelen utilizar como abreviatura los sistemas de numeración octal (en base 8), si el número de "bits" es múltiplo de 3, o hexadecimal (en base 16), si el número de "bits" es múltiplo de 4.

Una instrucción máquina se expresaría en estos códigos así:

Octal 22022317
Hexadecimal 4824CF

2.22.2 Segunda generación: Los *lenguajes simbólicos*, así mismo propios de la máquina, simplifican la escritura de las instrucciones y las hacen más legibles.

Lenguaje simbólico: Comparando el mismo programa, escrito en código hexadecimal y en código simbólico, comprendemos la inmensa ventaja que supuso utilizar el segundo.

Cada instrucción de la máquina se transforma en una única instrucción en código simbólico.
Pero, además, para mejorar la legibilidad del programa, el código simbólico introduce instrucciones adicionales, que no corresponden a ninguna instrucción de la máquina y que proporcionan información. Se llaman "seudo instrucciones".

El código simbólico puede parecer de difícil acceso, pero es más fácil de recordar e interpretar que el binario o el hexadecimal.

Los lenguajes simbólicos no resuelven definitivamente el problema de cómo programar un ordenador de la manera más sencilla posible.

Para utilizarlos, hay que conocer a fondo el microprocesador, los registros de trabajo de que dispone, la estructura de la memoria, y muchas cosas más.

Además, el código simbólico está demasiado ligado al microprocesador para que sea posible escribir programas independientes de la máquina en que van a ejecutarse.

2.22.3 Tercera generación: Los lenguajes de alto nivel sustituyen las instrucciones simbólicas por códigos independientes de la máquina, parecidas al lenguaje humano o al de las Matemáticas.

Lenguajes de alto nivel. La programación en el lenguaje de la máquina o en lenguaje simbólico tiene ciertas ventajas:

- Mayor adaptación al equipo.
- Posibilidad de obtener la máxima velocidad con mínimo uso de memoria.
- También tienen importantes inconvenientes:
- Imposibilidad de escribir código independiente de la máquina.
- Mayor dificultad en la programación y en la comprensión de los programas.

Por esta razón, a finales de los años 50 surgió un nuevo tipo de lenguajes que evitaban los inconvenientes, a costa de ceder un poco en las ventajas.

Estos lenguajes se llaman "de tercera generación" o "de alto nivel", en contraposición a los "de bajo nivel" o "de nivel próximo a la máquina".

Principales lenguajes de alto nivel

Basic
Logo
ALGOL

Pascal
Object Pascal
C
C#
C++
Python
Perl
ADA
Cobol
Java
Fortran
Modula2

Lenguajes funcionales

- *Lisp*
- *Haskell*

Por su importancia es apropiado mencionar al **lenguaje C**, ya que es uno de los lenguajes de la programación que ha sido pionero entre los demás, pues su uso es de propósito general, además de que ha sido estrechamente ligado con el sistema UNÍS, pues fue ahí donde fue desarrollado, pues los programas que corren están hechos en C. Por su fácil adaptación, este lenguaje no esta ligado a ningún sistema operativo, por su utilidad para escribir compiladores y diversos programas para diversas disciplinas.

Proporciona una variedad de tipos de datos que son fundamentalmente caracteres, enteros y de punto flotante. Proporciona el control de flujo que se requieren en programas bien estructurados en lo que se refiere a la agrupación de proposiciones, toma de decisiones (if-else), selección de casos de entre un conjunto de ellos (switch), condiciones de paro (while, for) y la terminación de ciclos (break).

Con el paso del tiempo y gracias a (ANSI) American National Standars Institute, se aprobó el estándar de dicho lenguaje a finales de 1988, el cambio ha resultado benéfico para los programadores ya que por la nueva sintaxis ha ayudado para declarar y definir funciones, por lo que la compilación de los programas resulta efectiva al momento de detectar errores causados por argumentos no validos. Y sobre todo la biblioteca que acompaña a este, pues especifica la entrada al mismo sistema operativo.

2.24.4 Cuarta generación: Se ha dado este nombre a ciertas herramientas que permiten construir aplicaciones sencillas combinando piezas prefabricadas. Hoy se piensa que estas herramientas no son, propiamente hablando, lenguajes. Algunos proponen reservar el nombre de cuarta generación para la programación orientada a objetos.

Programación orientada a objetos.

La programación procedimental clásica presenta ciertos problemas, que han ido haciéndose cada vez más graves, a medida que se construían aplicaciones y sistemas informáticos más complejos, entre los que destacan los siguientes:

- Modelo mental anómalo. Nuestra imagen del mundo se apoya en los seres, a los que asignamos nombres sustantivos, mientras la programación clásica se basa en el comportamiento, representado usualmente por verbos.

- Es difícil modificar y extender los programas, pues suele haber datos compartidos por varios subprogramas, que introducen interacciones ocultas entre ellos.

- Es difícil mantener los programas. Casi todos los sistemas informáticos grandes tienen errores ocultos, que no surgen a la luz hasta después de muchas horas de funcionamiento.

- Es difícil reutilizar los programas. Es prácticamente imposible aprovechar en una aplicación nueva las subrutinas que se diseñaron para otra.

La programación orientada a objetos (OOP, por las siglas inglesas de Object-Oriented Programming) es una nueva forma de programar que proliferó a partir de los años ochenta y trata de encontrar solución a estos problemas utilizando los siguientes conceptos:

- **Objetos**: Entidades complejas provistas de datos (propiedades, atributos) y comportamiento (funcionalidad, programas, métodos). Corresponden a los objetos reales del mundo que nos rodea.

- **Clases**: Conjuntos de objetos que comparten propiedades y comportamiento.

- **Herencia**: Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.

- **Encapsulamiento**: Cada objeto está aislado del exterior, es un módulo natural, y la aplicación entera se reduce a un agregado o rompecabezas de objetos. El aislamiento protege a los datos asociados a un objeto contra su modificación por quien no tenga derecho a acceder a ellos, eliminando efectos secundarios e interacciones.

- **Polimorfismo**: Programas diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, aunque el significado del programa varíe según el objeto al que se aplica.

La programación orientada a objetos introduce nuevos conceptos, que a veces no son más que nombres nuevos aplicados a conceptos antiguos, ya conocidos. Entre ellos destacan los siguientes:

- **Método**: Es un programa asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena mediante un "mensaje".

- **Mensaje**: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros.
- **Propiedad, atributo o variable**: Datos asociados a un objeto o a una clase de objetos.

En la programación orientada a objetos pura no deben utilizarse llamadas de subrutinas, únicamente mensajes.

Por ello, a veces recibe el nombre de programación sin CALL, igual que la programación estructurada se llama también programación sin GOTO.

Sin embargo, no todos los lenguajes orientados a objetos prohíben la instrucción CALL (o su equivalente), permitiendo realizar programación híbrida, procedimental y orientada a objetos a la vez.

Entre los lenguajes orientados a objetos destacan los siguientes:

- *Smalltalk*
- *C++*
- *Java*
- *Delphi*

Cada elemento del mundo real es un objeto, esto significa que en cada uno de ellos podemos reconocer un conjunto de atributos que son propios de cada objeto y que determinan sus características y su apariencia, además, podemos asociar a cada uno de esos objetos un conjunto de acciones que pueden realizar, lo que se conoce como las capacidades del objeto.

Esta idea en su esencia permite captar el mundo real de una manera muy cercana a como realmente es, por lo que no es difícil de asimilar y comprender,

por ser un concepto bastante intuitivo. Tales conceptos son: objeto, clase, abstracción, herencia, encapsulamiento y método.

Metodologías de OO.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares.

UML, en su versión 1.0, fue propuesto en enero de 1997. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1

Este documento fue aprobado por el OMG en noviembre de 1997. El OMG llama a este documento OMG UML versión 1.1.

En esta se tratan de establecer normas con las cuales se puedan definir el encapsulamiento y las relaciones de forma tal que se obtengan un mayor número de criterios para poder obtener al final un modelo de objetos adecuado.

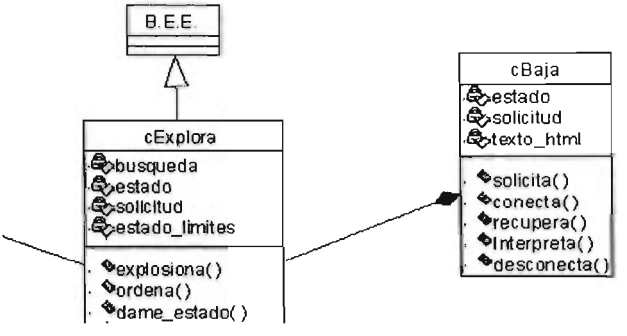
La metodología del software orientado a objetos consiste en:

- Saber el espacio del problema.
- Realizar una abstracción.
- Crear los objetos (espacio de la solución).
- Instanciarlos (esto es, traerlos a la vida).
- Dejarlos vivir (ellos ya saben lo que tienen que hacer) .

El lenguaje utilizado para representar dicha metodología, especialmente sus diagramas de clases y de interacción de objetos es UML (Lenguaje de Modelado Unificado).

Es un lenguaje de representación, utilizado para los diagramas generados durante el análisis y el diseño.

Pretende en un principio estabilizar el caos existente en las metodologías orientadas a objeto, así como la aparición de un modelo más rico, producto del intercambio de experiencias en las tres metodologías génesis de UML.



Modelo UML.

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. Resultando un lenguaje simple, común y ampliamente utilizable por usuarios de otras metodologías.

Capaz de representar los modelos que se obtienen a partir de la aplicación de cualquier metodología OO.

Además de ser una notación universal. El utilizar un lenguaje de modelado estándar le da un valor representativo. Lo que no quiere decir que se puedan utilizar otro tipo de metodologías porque existen distintos problemas que conducen a diferentes métodos de análisis y diseño.

Ofreciendo 9 diagramas para modelar dichos diseños:

Diagramas de Casos de Uso para modelar los procesos.
Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
Diagramas de Colaboración para modelar interacciones entre objetos.

Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.

Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
Diagramas de Componentes para modelar componentes.
Diagramas de Implementación para modelar la distribución del sistema.

2.22.5 Quinta generación: Se llama así a veces a los lenguajes de la inteligencia artificial, aunque con el fracaso del proyecto japonés de la quinta generación el nombre ha caído en desuso.

Inteligencia artificial.

Inteligencia Artificial (IA) es una de las áreas más fascinantes y con más retos de las Ciencias de la Computación, en su área de Ciencias Cognoscitivas. Nació como mero estudio filosófico y del razonamiento de la inteligencia humana, mezclada con la inquietud del hombre de imitar la naturaleza circundante (como volar y nadar), hasta inclusive querer imitarse a sí mismo.

Es una de las áreas de las ciencias computacionales encargadas de la creación de hardware y software que tenga comportamientos inteligentes

La I.A. tuvo su mayor auge a partir de la conferencia efectuada en el Darmouth

□ollage en 1956 (aquí nace la I.A.)

Estando presentes científicos como:
1.-John Mccarty: quien le dio el nombre a esta nueva área del conocimiento.
2.-Marvin Minisky: fundador del laboratorio de I.A. del Mit.
3.-Claude Shannon: de los laboratorios Bell de EE.UU.

En términos generales las eras por las que ha pasado la I.A. son las siguientes:

1. El inicio (1956-1965) poniéndose principal énfasis en la implementaron de juegos en el computador (ajedrez, damas etc.), Asi como en la demostración de teoremas.

2. La etapa oscura (1965-1970) aquí se apoya el entusiasmo por la I.A.

3. La etapa del renacimiento de la I.A. (1970-1975), iniciado en la Universidad de Stanford con el sistema experto medico Mycin (experto en enfermedades infecciosas de la sangre como la meningitis).
4. La etapa de las sociedades. (1975-1980) aquí se identifica la necesidad de trabajar en sociedad con profesionales en otras áreas del conocimiento.
5. La etapa de la comercialización de la I.A.

La IA ha explorado las distintas formas en que las computadoras podrían realizar las tareas que antes estaban reservadas a los seres humanos, como resolver problemas, planear a futuro, demostrar teoremas, jugar ajedrez, conversar en y entender un lenguaje, componer música, etc.

El hecho de que aún no haya conseguido reproducir un ser humano completo (o que eventualmente lo consiga) es de menor importancia que la evidencia de que ha mejorado nuestras habilidades para pensar y clarificar fenómenos de interés para la psicología y otras ramas de la ciencia.

CAPÍTULO 3

EVALUACIÓN DE HERRAMIENTAS

3.1 Software Libre

En los últimos 2 años el desarrollo del software libre ha venido creciendo, acaparando un mercado cada vez mayor que hace 5 años no se esperaba alcanzar en cuanto a velocidad, seguridad y precio se refiere.

Demostrando de esta manera la solidez de este software, su crecimiento y la aceptación ante todo por parte de la industria, además de que pueden adaptar utilerías a necesidades ya existentes y pueden ser modificadas de acuerdo a las necesidades de la empresa.

Ventajas y Desventajas del Software

Software	Ventajas	Desventajas
Licencia	Aplicaciones ya listas. Apoyo de fabricantes de hardware. Apoyo a las empresas.	Precios elevados. Restricciones de distribución. Dificultad de adaptar la aplicación a necesidades específicas.

Libre	Aplicaciones con código abierto, fácil de modificar y adaptar. Precio muy bajo o gratuito. Código relativamente sencillo garantizando la potabilidad. Parches generados en poco tiempo. Actualizaciones del software vía Internet. La velocidad de operación es mayor. El requerimiento de hardware es menos demandante. Se soporta una gran cantidad de tecnologías adicionales (PDF, Flash, FTP, etc.), lo que hace más flexible el producto final. El servidor puede incluir un cortafuegos para evitar intrusiones, además inmune a más del 99% de los virus en Internet.	Soporte técnico "certificado" más escaso, aunque el Calificado abunda aunque no haya papel que lo certifique. La interfaz y el tipo de administración del servidor es sustancialmente diferente a la que se realiza en Windows. Las actualizaciones requieren de conocer ciertos aspectos generales de la arquitectura, ya que en ocasiones es necesario compilar las aplicaciones. Configurar diversos servicios de red requieren de más tiempo que en Windows. Es necesario capacitar al personal o contratar personal calificado, lo que implica un incremento en los costos de la administración de personal.

El pago por licencias se reduce considerablemente o incluso a la mitad al obtenerlos de forma gratuita como es el caso de LINUX, por lo que el coste final de los equipos de cómputo disminuye notablemente al no pagar licencias por instalación de SO en cada equipo, como en el caso de Microsoft.

EL Software Libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

Un programa es software libre si los usuarios tienen todas estas libertades.

Un programa libre debe estar disponible para uso comercial, desarrollo comercial y distribución comercial.

El desarrollo comercial del software libre ha dejado de ser inusual; el software comercial libre es muy importante.

El Software Libre ofrece libertad

El Software Libre proporciona la libertad de: ejecutar el programa, para cualquier propósito; estudiar el funcionamiento del programa, y adaptarlo a sus necesidades.
Redistribuir copias, mejorar el programa, y poner sus mejoras a disposición del público, para beneficio de toda la comunidad.

3.1.1 Licencias de Software Libre

Las licencias bajo las cuales se rigen los términos de uso del software libre son la parte medular que determina la libertad o no de determinado programa, es por lo tanto muy importante el conocer y comprender su contenido.

El software libre para mostrar claramente qué debe cumplir un programa de software concreto para que se le considere software libre.

El ``Software Libre'' es un asunto de libertad, no de precio.

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades. El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias.
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

El software libre permite adaptaciones y cambios para elaborar publicaciones electrónicas utilizando software gratuito o adaptado para producir formatos que son leídos por software ampliamente disponibles. Generalmente es más difícil manejarlos y requiere de mayor aprendizaje. Un ejemplo, sería la utilización de cualquier procesador de textos para elaborar páginas html.

Otro ejemplo de lo anterior y quizá el más importante en la era del software libre es el sistema GNU (GNU es un acrónimo recursivo para ``GNU No es Unix'' y se pronuncia "gnu", tal y como se escribe). Variantes del sistema operativo GNU

que utilizan el núcleo Linux, son bastante utilizadas hoy día; aunque estos sistemas son frecuentemente referidos como ``Linux'', deberían llamarse más exactamente sistemas GNU/LINUX.

3.2 MS-DOS/Windows frente a Linux

La intención de esta sección es ofrecer una breve comparativa entre los sistemas operativos Windows y Linux. No se trata de una comparación exhaustiva, ni tampoco se entra profundamente en diferencias de rendimiento, estabilidad, etc.

Tanto Windows como Linux son sistemas multitarea, es decir, pueden ejecutar varios programas al mismo tiempo. Los sistemas UNIX, además, son multiusuario, lo que significa que muchos usuarios pueden estar usando a la vez la máquina sin influirse mutuamente. Cada usuario tiene unos privilegios que le permiten realizar ciertas cosas. Sólo el superusuario tiene el control absoluto sobre la máquina. En Windows, todos los programas tienen todos los derechos y pueden modificar, crear y eliminar cuantos objetos necesiten.

Los sistemas Unix están diseñados para funcionar adecuadamente en remoto. La red es una extensión natural al sistema, y todos los programas están pensados teniéndola en cuenta: la naturaleza monousuario de Windows hace que la red no esté tan bien integrada con el resto del sistema, funcionando bien como ordenador cliente, pero no tan bien como servidor.

Windows soporta nativamente la mayoría de los periféricos que se encuentran en los PCs actuales. En los casos en que un periférico no es soportado, el fabricante aporta los drivers necesarios. En Linux no puede haber drivers binarios, por lo que para que un periférico esté soportado, el fabricante debe liberar las especificaciones del mismo. Hay pocos fabricantes que hagan eso y, por tanto, el número de periféricos soportados por Linux es menor. Actualmente, este número está creciendo rápidamente, aunque sin llegar a los niveles de Windows.
En cambio, Linux es capaz de ejecutarse en muchas más arquitecturas. Windows sólo funciona en microprocesadores Intel basados en x86, mientras que Linux se

ejecuta en x86, Sparc, Alpha, Motorola 68000 (PowerPC, Macintosh), MIPS, ARM, Sparc64, etc.

Cuando hablamos de software como Linux, lo primero que pensamos es en "gratis", tanto porque se puede descargar sin costo de licencias, como porque en inglés se le llama free software, cuya connotación real es relativa a "libre", no a "gratis", motivo por el que el hispano parlante le llama a esta corriente software libre. Su principal característica no es la gratuidad, sino la libertad de modificarlo de acuerdo a nuestras necesidades.

Pero tienen un costo el cual se traduce en hardware donde trabajar, capacitación del personal para operarlo y soporte técnico que respalde la operación. Sin embargo, es necesario identificar que son los mismos requerimientos que tendríamos si utilizáramos Solaris, Windows 2000 o Mac OS X.

Hardware: Encontramos una gran variedad de servidores de marca donde correrlos, como la Series de IBM o PowerEdge de Dell por ejemplo, cuyos costos van desde los 2,800 USD hasta los 6,200 USD con procesadores desde Pentium III hasta Xeon.

Capacitación: Los costos de cursos intermedios por persona, dependiendo la empresa que los proporcione, del temario y el número de horas, van desde los $2,500.00 M.N. hasta los $9,000.00 M.N. por persona.

Soporte técnico: Los servicios de soporte sobre distribuciones específicas RedHat, Mandrake, SuSe teniendo en cuenta que se puede hacer a través de Internet como una opción, aunque existen otras posibilidades y el costo es bajo.

Un comparativo nos muestra los siguientes resultados sobre los costos de mantenimiento y administración:

- Solaris: ($561,520 USD)
- Windows: ($190,662 USD)

- Linux: ($74,475 USD)

Notemos como Linux no es "gratuito", ya que tiene un costo por los mismos conceptos que los otros, sin embargo, resulta un 40% más bajo que Windows. Este mismo estudio encontró que, dentro de su muestra, los administradores de redes Linux cobran casi tanto como sus contrapartes para Windows, pero se requiere menos personal.

En la teoría, el usuario final sólo requiere de un navegador de Internet y de una conexión que le permita conectarse al servidor, ya sea por dial-up (módem), LAN/WAN, enlaces de alta velocidad u otro, de acuerdo a sus propias necesidades.

Esto limita los costos inherentes a instalar una aplicación en cada equipo, lidiar con diferentes versiones de Windows y reducir la curva de aprendizaje de los usuarios (siempre y cuando, la interfaz esté bien diseñada).

Un elemento muy importante es lo referente a la seguridad, ya que una aplicación que esté en operación todo el tiempo y su información no esté desprotegida.

Al ser software libre, no existe el costo de las licencias, y una copia del sistema GNU/Linux puede instalarse en tantas computadoras como se necesite.

El siguiente es un análisis de costos operativos totales de compra y uso de una empresa en usar Gnu/Linux en comparación con plataformas Windows.

Análisis de costos respecto a Windows

Windows

Precio Proveedor
Norton Antivirus 2002 $ 49.95 Symantec
Microsoft Internet inf.(iis) Gratis Microsoft
Microsoft Windows 2000 adv. server $ 3,999.00 Microsoft

Windows XP Professional full ver. $ 299(por usuario)Microsoft

Comentarios:
Incluido con Microsoft NT y 2000 Viene con 25 Licencias de acceso de clientes (CALs). Cada CAL adicional cuesta $ 67 dólares.
Esta es una licencia por procesador.
El producto incluye SQL Server.
El producto incluye software de contrafuegos y servidor proxy.
Esta es una licencia por procesador.
Viene con 5 licencias de acceso para clientes (CALs). Cada CAL adicional cuesta $ 67 dólares.

Todos los precios están en dólares EE.UU. 19/04/02.

Análisis de costos respecto a Unix

Unix

Precio Proveedor
Cajas oficiales de Red Hat 7.2 $59.95
Mandrake 8.1 $55.00
o SUSE 7.3 $ 79.95

Incluido en la distribución LINUX (descarga gratuita)

Apache (Servidor Web)
Squid (servidor proxy)
Iptables (contrafuegos)
Sendmail o Postfix (servidores de correo)

Comentarios:
APACHE un servidor Web eficiente y extensible, usando por el 59 % de los servidores en Internet.

Sendmail es un potente y flexible servidor de correo que tiene el 80% de mercado de servidores de correo de Internet.
Open Office una suite ofimática completa, compatible con Microsoft office, que funciona sobre linux, Solaris y Windows.

Todas las cifras son en dólares EE.UU. 19/04/02

CAPÍTULO 4

DESARROLLO DEL PORTAL DE BIBLIOTECAS

(PUESTA EN MARCHA DE LA APLICACIÓN)

4.1 DEFINICION DEL PROBLEMA.

La problemática que enfrenta actualmente la Biblioteca del Anexo de Facultad de Ingeniería (Enrique Rivero Borrel), en cuanto a su material no librario, es que no cuenta con un método eficaz para el almacenamiento de la información, es decir, su información no se encuentre documentada.

Existe solo una persona encargada del control del material no librario, es la única que tiene conocimiento de su ordenamiento, pues la mayoría del material se cataloga con una clasificación interna y no es de conocimiento a los usuarios.

Además no existe un control del material adquirido, como resultado de esta situación la mayoría de los usuarios desconoce la existencia y disposición del mismo, pues al realizar una consulta en el sistema de búsqueda de la biblioteca se obtiene como resultado únicamente material librario.

Debido a la problemática interna, no hay un compromiso de diseñar o dar soluciones adecuadas y dar a conocer esta información que es valiosa, que ha pasado desapercibida y que al final no satisface al usuario ni al bibliotecólogo.

4.2 SOLUCION PROPUESTA.

1.-Existen diversas alternativas que son principalmente la compra de un software adecuado para satisfacer esta necesidad.

2.-Dar a conocer a través de la página de la biblioteca de la Facultad de Ingeniería la existencia de este tipo de material, no así de su consulta.

3.-Realizar visitas guiadas a la biblioteca para dar a conocer la existencia de este material.

4.-Crear un catálogo o fichas bibliográficas para poder hacer consulta de los mismos.

Al realizar el análisis en el punto No.1, la compra de un software requiere de un presupuesto elevado y que continuamente se tendrá que estar actualizando.

El punto No.2, no basta solo con publicar vía Web la existencia del material sino poderlo consultar.

Las visitas guiadas en el punto No.3 y que por lo general son voluntarias, han sido de un gran desinterés por parte de la comunidad de esta facultad.

El punto No.4 quizá hubiera funcionado en su momento años atrás, pero ahora es dar paso a algo más cómodo, rápido y accesible a cualquier persona.

Es por ello que la opción que presenta mayores beneficios es el desarrollo de un sistema en el cual se lleven a cabo consultas, para el usuario y por la parte del bibliotecólogo poder realizar actualizaciones, inserciones y bajas de material. Haciendo uso de la Internet como medio de comunicación.

4.3 ALCANCES.

El sistema proporcionara a los usuarios información acerca del material no librario existente y a su vez la disponibilidad de dicho material, esto es tanto para consulta interna como para consulta externa.

El sistema realizara búsquedas por autor, titulo, tema y proporcionará los datos importantes del mismo.

A nivel administrativo el personal encargado de dicho sistema tendrá privilegios para hacer altas, bajas y actualización del material.

Cuando se realice alguna modificación o borrado el administrador deberá conocer el título, la clasificación y con ello se busca proporcionar la información completa de dicho material para los fines que convenga.

El sistema mostrará sólo la información que fue proporcionada por el personal de la biblioteca. Todo este material se encuentra alojado en la biblioteca del Anexo.

4.4 Análisis de herramientas.

Tomando en cuenta el estudio de los requerimientos de nuestro sistema podemos llevar a cabo un análisis de las herramientas que nos proporcionarán un funcionamiento y rendimiento óptimo de nuestro sistema.
Las herramientas que se utilizarán serán para el almacenamiento, presentación y el medio por el cual se procesará la información entre el usuario y la biblioteca.

- Como medio de almacenamiento utilizaremos el manejador de Bases de Datos Sybase.
- El medio para la presentación de la información para el usuario será HTML
- Para procesar la información del almacenamiento de los datos y el usuario utilizaremos PHP.

Sybase

El IIMAS (Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas) hace uso del manejador de bases de datos Sybase pues ofrece flexibilidad, rendimiento y seguridad que precisan las nuevas aplicaciones. En comparación con Postgresql que consume bastantes recursos y carga más el sistema haciéndolo más lento, por su parte Mysql aunque fue diseñado para consumir menos recursos y ser más veloz, no considera las llaves foráneas e ignora la integridad referencial, por otro lado Oracle presenta mejores herramientas pero a su vez un mayor consumo de recursos y mayores costos comparado con los anteriores DBMS, por lo que no representa una opción viable en nuestro caso, ya que el total de registros que se manejaron es pequeño.

En nuestro proyecto usamos **ERWIN**, una herramienta CASE para optimizar el diseño y acelerar la implantación de aplicaciones ligadas a las bases de datos, permitiendo, además, reducir errores y aumentar la calidad.

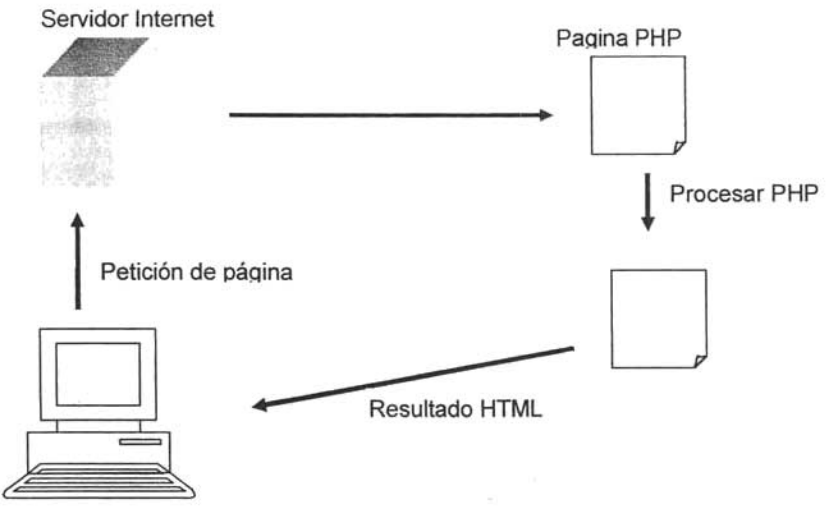
Las herramientas **CASE** (Ingeniería Asistida por Computadora) tienen el objetivo de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema. Los aspectos que tiene una herramienta CASE son los siguientes:

- Un diccionario de datos para almacenar información sobre los datos de la aplicación de bases de datos.
- Herramientas de diseño para dar apoyo al análisis de datos.
- Herramientas que permitan desarrollar el modelo de datos corporativo, así como los esquemas conceptual y lógico.
- Herramientas para desarrollar una interfaz de usuario.

El uso de las herramientas CASE puede mejorar la productividad en el desarrollo de una aplicación de bases de datos.
La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información.
Para el lenguaje de desarrollo del DBMS usaremos SQL y la plataforma será en Windows y la arquitectura a usar es la de tres capas de cliente/servidor.

Para el desarrollo de las páginas Web usaremos PHP y HTML. PHP es un lenguaje de programación soportado por HTML. La sintaxis esta heredada de C, Java y Perl. Orientado para la creación de páginas Web, permitiendo crear paginas dinámicamente generadas de forma rápida.

Es un lenguaje de programación de estilo clásico con sentencias, variables, etc., esta más cercano a JavaScript o a C, pero a diferencia de los anteriores PHP se ejecuta en el servidor, permitiendo acceder a los recursos que tenga el servidor como por ejemplo una base de datos.



El programa PHP es ejecutado en el servidor y el resultado enviado al navegador.

El resultado es normalmente una pagina HTML. Al ser un lenguaje que se ejecuta en el servidor no es necesario que el navegador lo soporte, ya que es independiente del navegador, sin embargo, para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

Otra de las características importantes es que PHP nos provee de una extensa gama de funciones de acceso a archivos.

Quizá la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Realizar 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 (OCI7 and OCI8)
dBase	InterBase	PostgreSQL
Empress	FrontBase	Solid
FilePro	mSQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Diferentes Bases de Datos

HTML: Como mencionamos en el capitulo II, es de gran importancia el contar con esta herramienta para el diseño de la interfaz del usuario y el administrador que hará conexión con el servidor donde se encuentra alojada la base de datos.

La característica de HTML por ser un lenguaje estándar, es que tiene una gran cantidad de herramientas para lograr el buen desempeño del portal, con base en formularios, ligas, etc.

4.5 METODOLOGIA USADA.

La metodología utilizada para el desarrollo del sistema fue la de desarrollo incremental.
Ya que en este tipo de método el sistema es particionado en subsistemas de acuerdo con su funcionalidad. Las versiones se definen comenzando con un subsistema funcional pequeño y agregando funcionalidad con cada nueva versión.

- Comienza sobre una versión temprana, aun si se han omitido funcionalidades. El proceso de entrenamiento permite que los desarrolladores observen como se ejecutan ciertas funciones, sugiriendo los cambios que las refuercen para versiones posteriores. De esta forma el desarrollador puede dar respuesta a las necesidades de los usuarios.

- Permite la creación de módulos para funcionalidades que nunca antes se habían ofrecido.

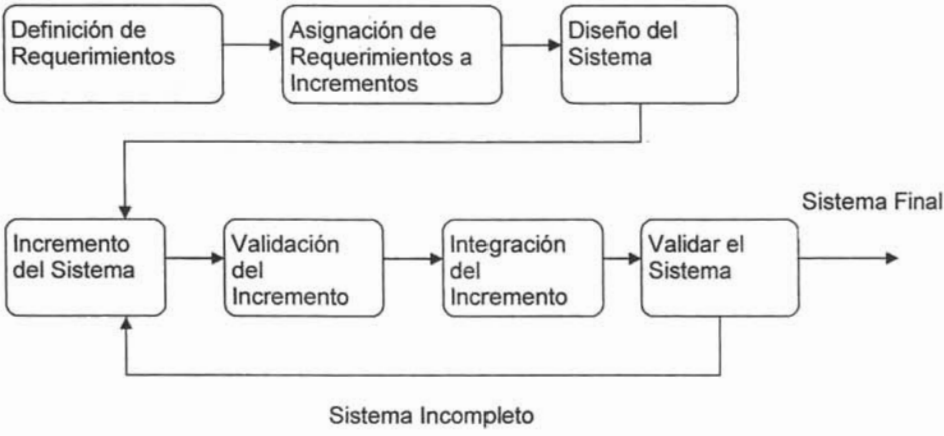
- Las versiones frecuentes permiten que los desarrolladores arreglen problemas no anticipados de manera global y rápida, a medida que son informados desde el sistema operacional.

- El desarrollo puede centrarse en diferentes áreas de especialización con las diferentes versiones. Por ejemplo una versión puede cambiar la modalidad de la interfaz del sistema de la actual por comandos al modo grafico (apuntar y hacer clic), utilizando la experiencia de los especialistas en interfaces de

usuarios; otra versión puede enfocarse sobre la manera del rendimiento del sistema.

En vez de entregar el sistema como una única entrega, el desarrollo y la entrega son descompuestos en incrementos con cada incremento entregando una parte de la funcionalidad requerida.

A los requerimientos de usuario se les da prioridad y los requerimientos de alta prioridad son incluidos en los primeros incrementos.



Metodología Incremental (Diagrama).

- Definir requerimientos a grandes rasgos: Es importante desde un inicio el saber los requerimientos que el cliente necesita. Es decir, el tipo de sistema que se va a usar y por otro lado si no habrá algún inconveniente al hacerlo.

- Para nuestro sistema se tomaron en cuenta los puntos anteriores como son el tipo de servicio que va a proporcionar y los posibles inconvenientes que se llegaron a dar, algunos de ellos son en cuanto a los datos reales del material que no están completos.

- Asignar requerimientos a incrementos: La asignación de estos requerimientos a cada una de las partes del sistema que en este caso se dividen en módulos para que al final puedan ser modificados por separado y no afecten al sistema en su totalidad.

  Cada módulo como son consultas, altas, bajas y actualizaciones se contemplaron en este apartado.

- Diseñar la arquitectura del sistema: Una de las partes más importantes es esta, realizada por los diseñadores. Que involucra el diseño y la programación tanto de la base como de la página Web.
  En caso dado de que en alguno de ellos exista un problema el administrador solo se enfocara en el que tenga el problema.

  El diseño del sistema se crea para satisfacer los requerimientos especificados. En este diseño del sistema se describe solamente la apariencia y la funcionalidad. Este diseño es revisado por el cliente. Cuando sea aprobado, se usa el diseño del sistema global para generar los diseños de los programas individuales involucrados.

- Desarrollar incremento del sistema: ya teniendo el sistema podrá ser mejorado y agregar o quitar según sea el caso de la modificación. Esto es probar cada una de las partes de la base de datos en cuanto a sus autores, editoriales, clasificación, etc. Y la parte de la página Web para las consultas, altas, bajas, etc. La primera fase se denomina prueba unitaria o prueba de módulos.

- Validar incremento: Una vez hecho lo anterior este incremento tendrá que ser validado por el cliente, de ser así solo se harán las acotaciones

correspondientes. Pues es la parte fundamental para saber si el diseño cumple con los requerimientos establecidos.

- Integrar incremento: Después de que él módulo ha sido incrementado, se integrara a los demás para hacer las pruebas. Una vez que se demuestra que cada pieza trabaja como se desea, es decir, si se realiza una consulta esta deberá mostrar solo la consulta, esto para el caso del usuario. Posteriormente se unen y se asegura que trabajan correctamente con las otras.

- Validar sistema: La fase final de prueba, denominada prueba del sistema, implica la prueba del sistema completo para asegurar que las funciones y las interacciones especificadas inicialmente se han implementado correctamente. En esta fase se compara con los requerimientos especificados, el desarrollador, el cliente y los usuarios comprueban que el sistema cumple con el objetivo. Esto es tanto la interfaz con el usuario y la base de datos realizan lo especificado.

- Sistema incompleto: Si por algún motivo presenta fallas y resulta incompleto se tendrá que revisar nuevamente y se regresa a la etapa de incremento del sistema, hasta que pueda ser aprobado. Por ejemplo el usuario al realizar una consulta esta arroja datos que solo puede revisar el administrador o si uno de los módulos falla o si la pagina no se muestra correctamente.

- Sistema final: Por último se entrega el producto final. A medida que se utiliza se descubren problemas y discrepancias. El cliente asume la responsabilidad por el sistema una vez entregado.

Si los módulos que comprenden todo el sistema en su totalidad funcionan correctamente y se hicieron las revisiones ya antes citadas, el sistema final se entrega con la solidez de que el cliente ha hecho las pruebas necesarias y se le ha instruido para su buen funcionamiento.

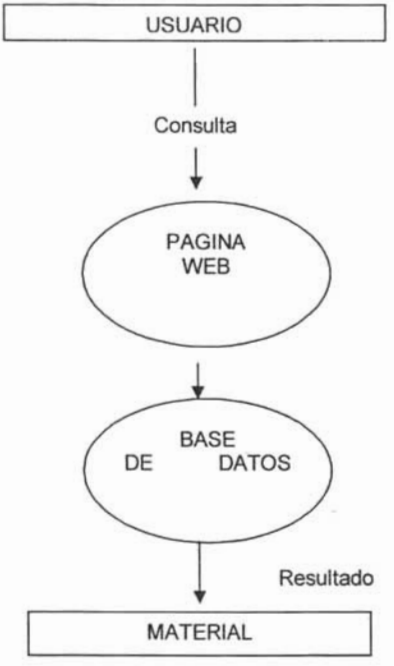
4.6 VENTAJAS.

- Cosas que tienen valor para el cliente pueden ser entregadas con cada incremento para que la funcionalidad del sistema esté disponible lo más rápido posible.
- Incrementos iniciales funcionan como un prototipo para ayudar a obtener los requerimientos de incrementos futuros.
- Más bajo riesgo de fracaso del sistema entero.
- Los servicios del sistema con la más alta prioridad tienden a recibir más pruebas.

4.7 CASOS DE USO.

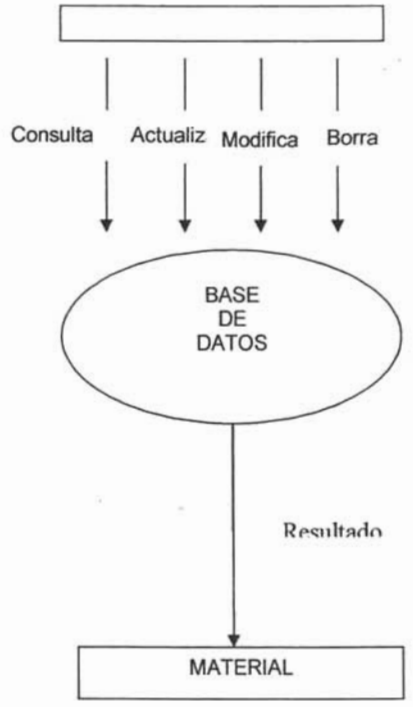
El usuario al realizar una búsqueda en la biblioteca sigue los siguientes pasos:

1. Realiza una consulta para la búsqueda de algún material en este caso material no librario.
2. La consulta es enviada a través de la página Web y esta a su vez se conecta a la base de datos donde se encuentra la información.
3. La base de datos realiza su trabajo y encuentra el material solicitado.
4. El resultado de la búsqueda es enviado al usuario.

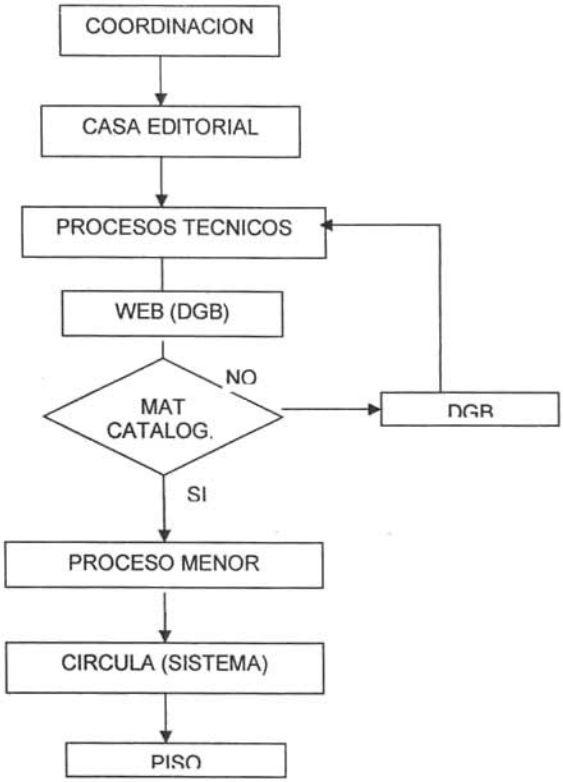


Para el caso del Administrador, tiene más privilegios para poder manipular la base de datos.

1. El Administrador del sistema se conecta directamente a la base de datos y hace ya sea una modificación, una consulta, una actualización o un borrado de algún dato.
2. Este proceso es invisible para el usuario.
3. Al hacerse estas modificaciones a la base no provocan ningún problema en el sistema.



A continuación describiremos los departamentos que forman parte de la biblioteca del Anexo de la Faculta de Ingeniería. El siguiente diagrama muestra el proceso que tiene que seguir el material para llegar a su destino final.



Dentro de la Coordinación se encuentra el área de adquisición que se encarga de ponerse en contacto con las casas editoriales y solicitar el material requerido. A su vez la Casa editorial de acuerdo al presupuesto asignado por la FI para la biblioteca, al evaluar este presupuesto envía el material a la biblioteca.

Una vez recibido el material es llevado al departamento de Procesos Técnicos en donde el material es clasificado de acuerdo a la clasificación establecida en la Dirección General de Bibliotecas.

Los cargos remotos se hacen vía Web hacia la DGB, en Procesos Técnicos se hacen los procesos menores. En este mismo departamento se preparan para ser etiquetados, etc.

Todo el material es cargado en el sistema llamado CIRCULA que es administrado por USECAD (Unidad de Servicios de Computo Académico).

Tras realizar esta clasificación el material pasa a la estantería.

En caso de que el material no este clasificado se detiene en el departamento de procesos técnicos hasta que la DGB le envié la adquisición y clasificación del mismo.

Se colocan y se hacen los cargos en el área de préstamo, cuando no esta cargado esta inventariado.

Para poder ser consultados (en el interior de la sala) por el usuario es necesario que presenten una credencial y llenar una forma.

Las ventajas que se obtienen con nuestro sistema son los siguientes:

- Llevar el control del material de una manera formal, es decir, el material no tendrá que ser llenado en un cuaderno.
- Facilidad de crecimiento y mantenimiento de la base de datos.
- Búsqueda más efectiva y rápida del material.
- Actualización del material adquirido.
- Hacer saber al usuario de este tipo de material.

4.8 Modelo de datos.

Definición del modelo de datos: Un modelo de datos es una combinación de tres componentes:

❖ Una colección de estructuras de datos (los bloques constructores de cualquier base de datos que conforman el modelo).

❖ Una colección de operadores o reglas de inferencia, los cuales pueden ser aplicados a cualquier instancia de los tipos de datos listados en (1), para

consultar o derivar datos de cualquier parte de estas estructuras en cualquier combinación deseada.

- Una colección de reglas generales de integridad, las cuales explícita o implícitamente definen un conjunto de estados consistentes; estas reglas algunas veces son expresadas como reglas de insertar-actualizar-borrar.

Modelo: Es una representación de la realidad que contiene las características generales de algo que se va a realizar. En base de datos, esta representación la elaboramos de forma gráfica.

Modelo de Datos: Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

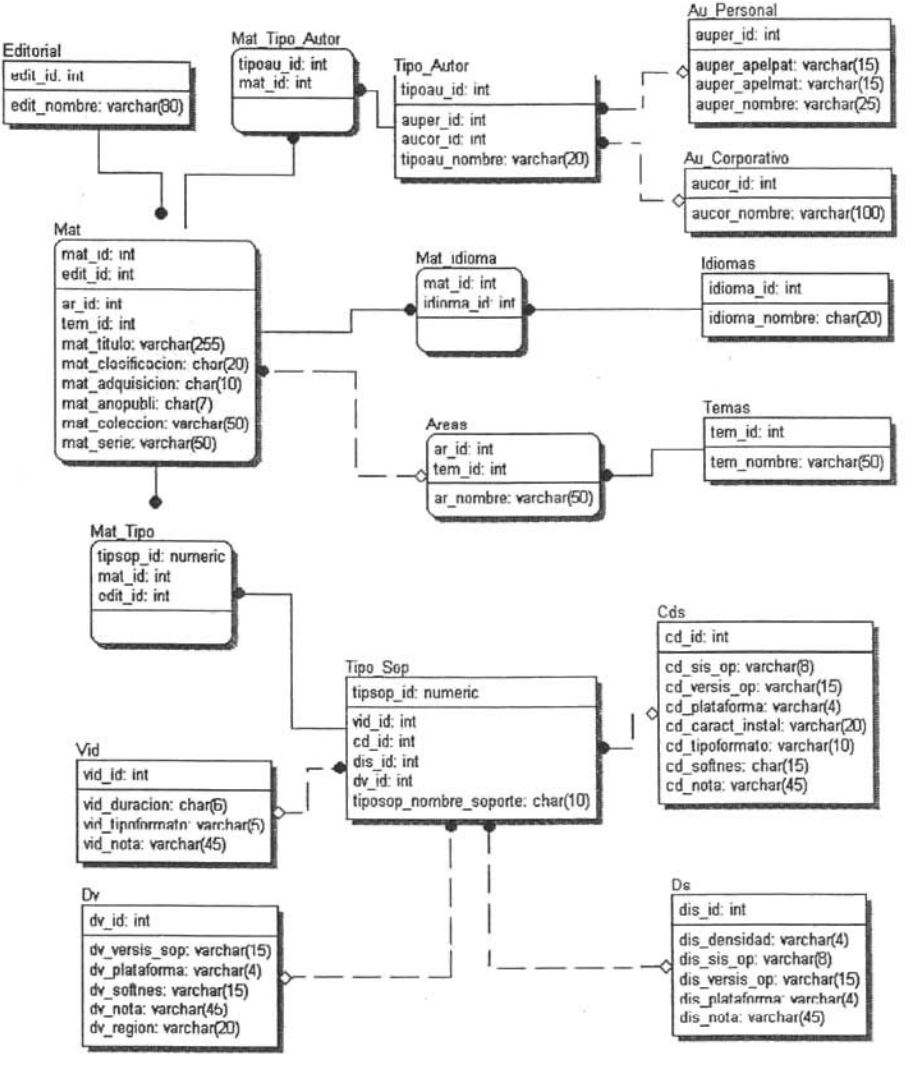
El siguiente es el diseño del modelo de la base de datos, que es una parte fundamental del sistema, ya que él contendrá la información requerida para el desarrollo del portal.

Para esta parte fue necesario hacer un exhaustivo análisis, de los datos para dicho modelado.

Después de obtener la información para la base de datos se formaron las entidades como sigue:

Las relaciones entre cada entidad fueron determinantes para la normalización eficaz de los datos y así poder tener un buen rendimiento de la base. Obteniendo las respectivas llaves primarias y foráneas.

4.9 Modelo Entidad- Relación.



4.10 Diccionario de Datos.

TABLA: **Au_Personal**

Autor es una tabla en la cual se establecen los campos que comprende las características de Autor Personal.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
au_id	Identificador del Autor	int	4	Not Null	✓	
auper_apel pat	Apellido Paterno	varchar	15	Not Null		
auper_apel mat	Apellido Materno	varchar	15	Null		
auper_nom bre	Nombre del Autor	varchar	25	Not Null		

TABLA: **Au_Corporativo**

Esta tabla nos hace referencia a los Autores Corporativos que pueden existir en el material en cuestión.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
aucor_id	Identificador del Autor Corporativo	int	4	Not Null	✓	
aucor_nom bre	Nombre del Autor Corporativo	varchar	100	Not Null		

TABLA: **Tipo_Autor**

En esta tabla se relacionan los identificadores de las tablas del Autor y se ingresa el Tipo_Autor, correspondiente al material.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
tipoau_id	Identificador: Tipo_Autor	Int	4	Not Null	✓	
auper_id	Identificador: tabla Au_Personal	int	4	Null		✓
auper_apel mat	Identificador: tabla Au_Corporativo	Int	4	Null		✓
auper_nom bre	Nombre del Tipo del Autor	varchar	20	Not Null		

TABLA: **Mat_Tipo_Autor**

La tabla Mat_Tipo_Autor hace referencia a la relación que existe entre el tipo material, su autor y la editorial del mismo, relacionando así los datos ingresados.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
tipoau_id	Pertenece a la tabla Tipo_Autor	int	4	Not Null		✓
mat_id	Pertenece a la tabla Mat	int	4	Not Null		✓
edit_id	Pertenece a la tabla Editorial	int	4	Not Null		✓

TABLA: **Idiomas**

La tabla Idioma comprende el tipo de idiomas en el cual esta disponible el material.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
idioma_id	identificador Idioma	int	4	Not Null	✓	
idioma_nombre	Nombre del idioma	char	20	Not Null		

TABLA: **Mat_idioma**

Mat_idioma relaciona los datos registrados de la tabla Mat e idioma, almacenando así la información que se contenga en ellas.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
mat_id	Pertenece a la tabla Mat	int	4	Not Null		✓
edit_id	Pertenece a la tabla Editorial	int	4	Not Null		✓
idioma_id	Pertenece a la tabla Idioma	int	4	Not Null		✓

TABLA: **Temas**

En la tabla Temas se almacenaran todos los temas generales en los cuales esta el material.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
tema_id	Identificador Tema	int	4	Not Null	✓	
tema_nombre	Nombre del tema	varchar	50	Not Null		

TABLA: **Áreas**

En la tabla Áreas se hace una relación entre la tabla temas y el nombre del área específica en la que están los materiales existentes.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
area_id	Identificador de la tabla Áreas	int	4	Not Null	✓	
tema_id	Pertenece a la tabla Tema	int	4	Not Null		✓
area_nombre	Nombre del área	varchar	50	Not Null		

TABLA: **Mat**

La tabla Mat establece las características más generales y esenciales de cada material digital.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
mat_id	Identificador de la tabla Mat	int	4	Not Null	✓	
edit_id	Pertenece a la tabla Editorial	int	4	Not Null		✓
tema_id	Pertenece a la tabla Temas	int	4	Null		✓
area_id	Pertenece a la tabla Áreas	int	4	Null		✓
mat_titlulo	Titulo del material	varchar	255	Not Null		
mat_clasificacion	Clasificación del material	char	20	Not Null		
mat_adquisición	Adquisición del material	char	10	Not Null		
mat_anopubli	Año de publicación del material	char	7	Not Null		
mat_coleccion	Colección del material	varchar	50	Not Null		
mat_serie	Serie del material	varchar	50	Not Null		

TABLA: **Editorial**

La tabla Editorial almacenara el nombre de la editorial del material digital.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
edit_id	Identificador de la tabla Editorial	int	4	Not Null	✓	
edit_nombre	Nombre de la editorial	varchar	80	Not Null		

TABLA: **Cds**

La tabla CD es uno de los formatos en la cual esta el material digital, cada uno de los campos es cuestión forman parte de las características que son el tipo de formato en la cual se podrá acceder este tipo de material.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
cd_id	Identificador de la tabla Cds	int	4	Not Null	✓	
cd_sis_op	Sistema Operativo	varchar	10	Not Null		
cd_versis_op	Versión del sistema operativo	varchar	20	Not Null		
cd_plataforma	Plataforma del para el Cd	varchar	5	Not Null		
cd_carac_instal	Caract. De instalación	varchar	20	Not Null		
cd_formato	Tipo de formato de	varchar	10	Not Null		
cd_softnes	Software necesario	char	15	Not Null		
cd_nota	Descripciones de disponibilidad	varchar	45	Not Null		

TABLA: **Ds**

La tabla Ds contiene información que al igual que la tabla Cds, nos presenta las características más generales en las cuales el material digital se puede ver.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
dis_id	Identificador de la tabla Ds	int	4	Not Null	✓	
dis_densidad	Densidad del ds	varchar	4	Not Null		
dis_sis_op	Sistema Operativo	varchar	8	Not Null		
dis_versis_op	Versión del sistema operativo	varchar	15	Not Null		
dis_plataforma	Plataforma para el ds	varchar	4	Not Null		
dis_nota	Descripciones de disponibilidad	varchar	45	Not Null		

TABLA: **Dv**

La tabla Dv es un tipo de formato la cual tiene características generales en las cuales se presenta un tipo de material.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
dv_id	Identificador del dv	int	4	Not Null	✓	
dv_versis_op	Versión del sistema operativo	varchar	15	Not Null		
dv_plataforma	Plataforma para el dv	varchar	4	Not Null		
dv_softnes	Software necesario	varchar	15	Not Null		
dv_nota	Descripciones de disponibilidad	varchar	45	Not Null		
dv_region	Región en la puede ser leído el soporte	varchar	20	Not Null		

TABLA: **Vid**

La tabla Vid es un tipo de formato en el cual están contenidas características y que por medio de ellas podemos ver el contenido del material digital en cuestión.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
vid_id	Identificador vid	int	4	Not Null	✓	
vid_duracion	Duración de vid	char	6	Not Null		
vid_tipoformato	Tipo de formato de video	varchar	5	Not Null		

vid_nota	Descripciones de disponibilidad	varchar	45	Not Null	

TABLA: **Tipo_Sop**

La tabla Tipo_Sop relaciona y almacena los tipos de soportes y el tipo de cada uno de ellos.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
tipsop_id	Identificador de la tabla tipo_sop	tinyint	1	Not Null	✓	
tiposop_nombre_soporte	Nombre del tipo de soporte	int	char(10)	Not Null		
vid_id	Pertenece a la tabla Vid	int	4	Null		✓
dv_id	Pertenece a la tabla Dv	int	4	Null		✓
dis_id	Pertenece a la tabla Ds	int	4	Null		✓
cd_id	Pertenece a la tabla Cds	int	4	Null		✓

TABLA: **Mat_Tipo_Sop**

En esta tabla se establece la relación de los registros del material con el tipo de soporte.

Atributo	Descripción	Tipo	Longitud	Nulos	Llave Primaria	Llave Externa
tipsop_id	Pertenece a la tabla Tipo_sop	tinyint	1	Not Null		✓
mat_id	Pertenece a la tabla Mat	int	4	Not Null		✓
edit_id	Pertenece a la tabla Editorial	int	4	Not Null		✓

4.11 DESCRIPCION GENERAL DE LOS MODULOS DEL SISTEMA

Módulo de Presentación:

Inicio: Da la bienvenida a los usuarios.

Módulo de Búsquedas:

Búsqueda Sencilla: Proporciona las búsquedas por Titulo y Autor Alfabéticamente.

Búsqueda Avanzada: Proporciona las búsquedas por Titulo, Autor y Tema General.

Módulo de Administración:

Admin.: Área para el Administrador, en el cual el Bibliotecólogo puede realizar las operaciones de alta, baja y actualización del material.

Módulo de Enlaces:

FI: Enlace a la página de Facultad de Ingeniería.

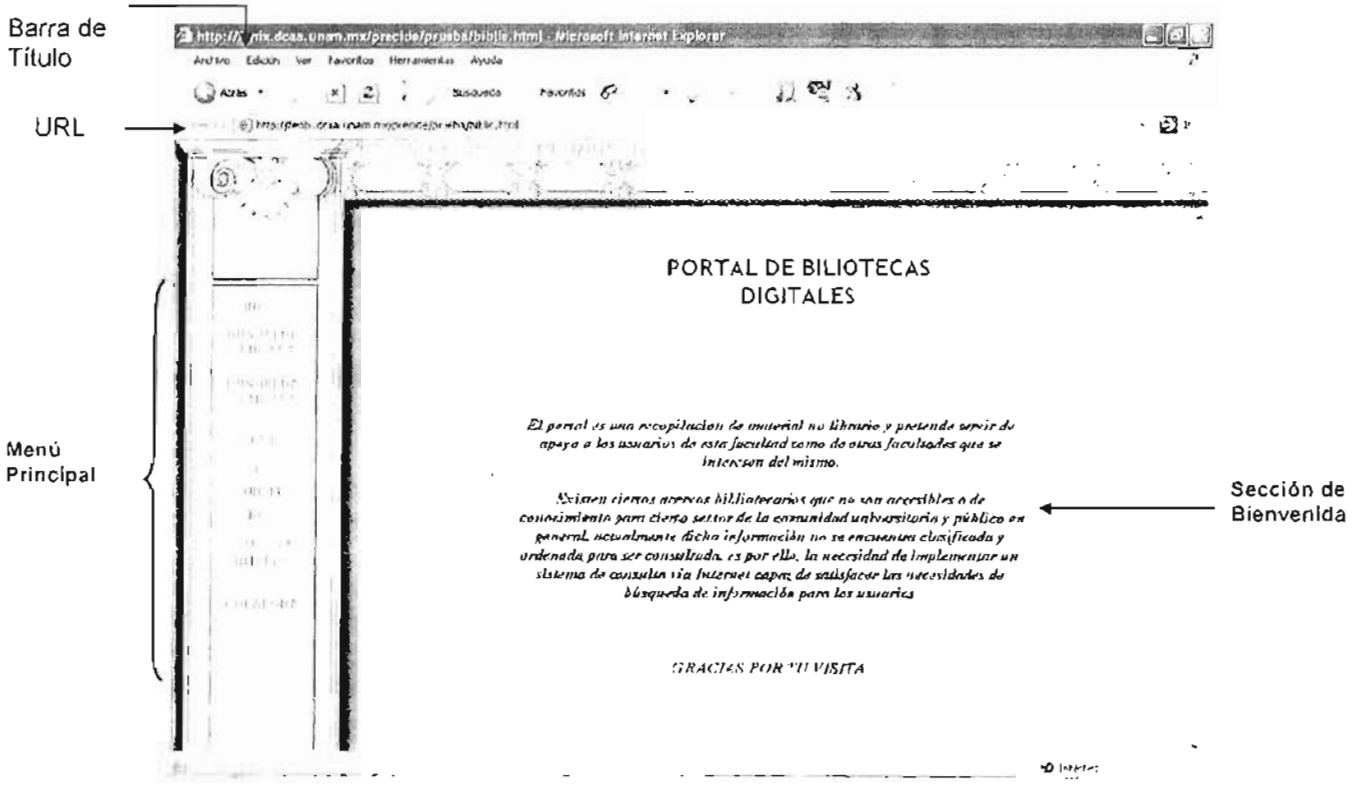
UNAM: Enlace a la página de la Universidad Nacional Autónoma de México.

DGB: Enlace a la página de la Dirección General de Bibliotecas.

Sitios de interés: Para futuros enlaces.

Colabora: Comentarios o sugerencias del Sistema.

PAGINA PRINCIPAL DEL PORTAL



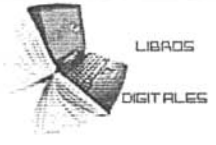
Barra de título: muestra la ruta donde se encuentra localizado el portal.

URL: aquí se introduce la dirección electrónica donde esta localizado el sistema.

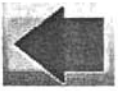
Menú principal: son las opciones que el usuario puede elegir.

Sección de bienvenida: ésta es una breve explicación de lo que ofrece a los usuarios este sistema.

CONTROLES DEL SISTEMA

Ligas: Permite ir a la información requerida a través del sitio.	A - B - C - D
Cuadros de edición de texto: Registro de información que el usuario necesita	TITULO
Cuadros de selección: Esta tiene por objeto elegir una opción a realizarse	ALTAS
Botones: Realiza una opción en particular.	Enviar
Imágenes con liga 1: Esta nos llevará a consultar algún autor en especial.	
Imágenes con liga 2: Esta nos llevará A consultar algún título en especial.	

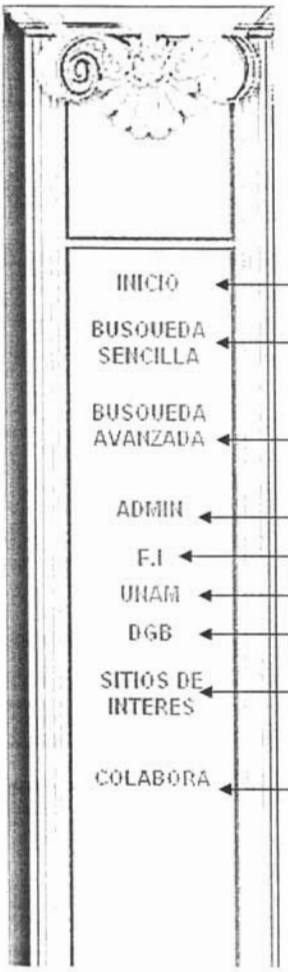
FUNCIONAMIENTO DE BOTONES

BOTON	FUNCION
	Regresa a la página principal, es decir, la de inicio.
	Después de ver alguna página con la información permite regresa a la página previa.
Enviar consulta	Envía al servidor una petición de consulta.
Enviar	Envía al servidor una petición.
Enviar datos	Envía al servidor una petición por parte del administrador.
Borrar	Borra los datos introducidos en los campos.
	Permite regresar a la página anterior.

MENSAJES DE RESULTADOS

Dentro de las pantallas que componen la página del administrador se encuentran mensajes y dependerán del tipo de opción que se quiera realizar, esto es una alta, una actualización, una baja.

ELEMENTOS DEL MENU PRINCIPAL



Nos muestra la página principal de inicio del portal.

Muestra los dos tipos de búsqueda del sistema, tanto por autor como por título.

Nos llevará a realizar una búsqueda más específica, ya sea por autor título o tema o una combinación de las anteriores.

Parte importante de esta liga, mandara al sitio del administrador quien tendrá todos los privilegios sobre la base de datos.

Estas son ligas de interés para acceder a sitios de la UNAM.

Desde aquí podrás enviar tus comentarios acerca del sitio.

Para cualquier usuario que requiera saber de la existencia de algún material podrá realizar la búsqueda sencilla, la cual se realizará a través de un glosario de tipo alfabético.

BUSQUEDA SENCILLA POR AUTOR

Seleccione la letra del comienzo del apellido del autor para ir directamente a dicha seccion

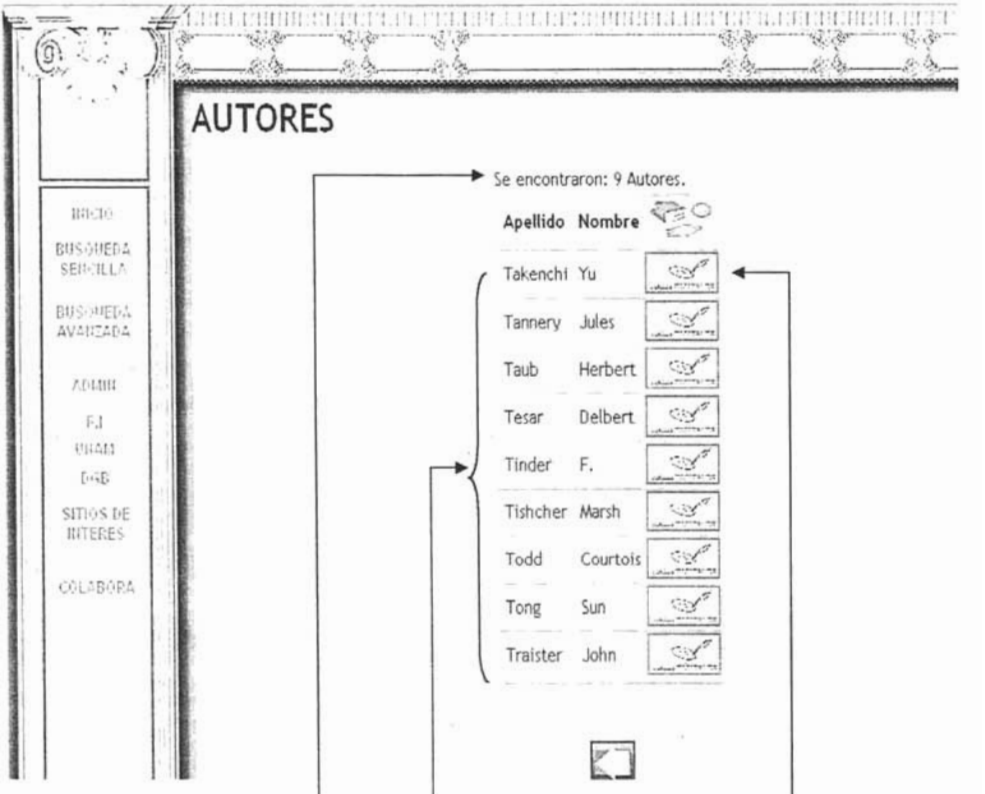
A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - R - S - T - U - V - W - Y - Z

Si la búsqueda es por autor

Seleccione la letra para ir directamente al titulo

A - B - K - T - U - V - W - Y - Z

En el ejemplo si la búsqueda es por autor desplegara el siguiente resultado

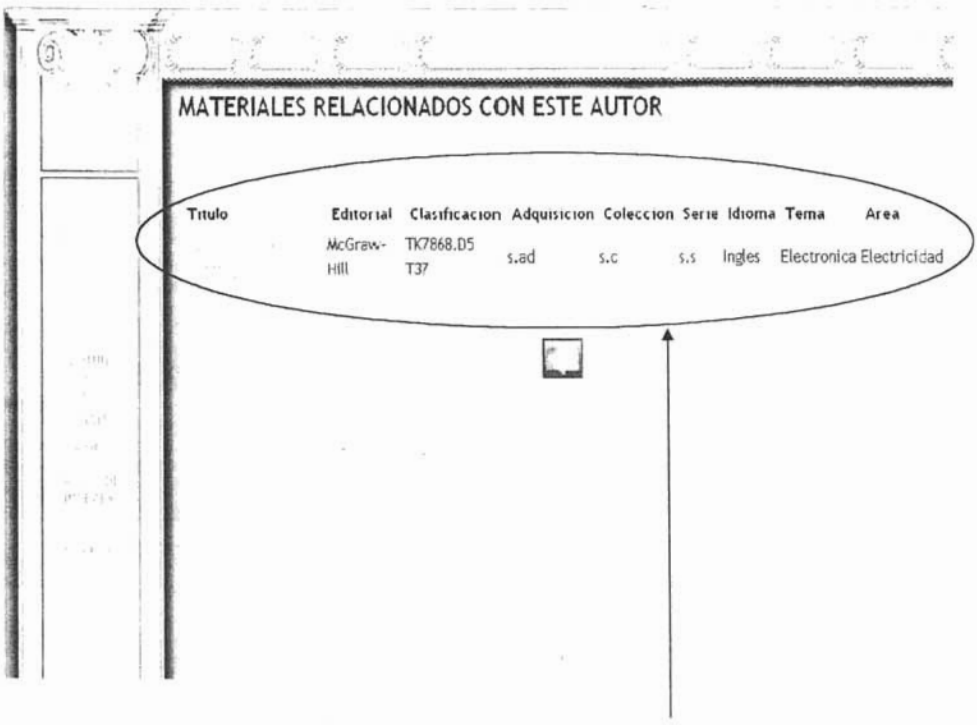


Nos muestra el número de resultados encontrados sobre dicha petición

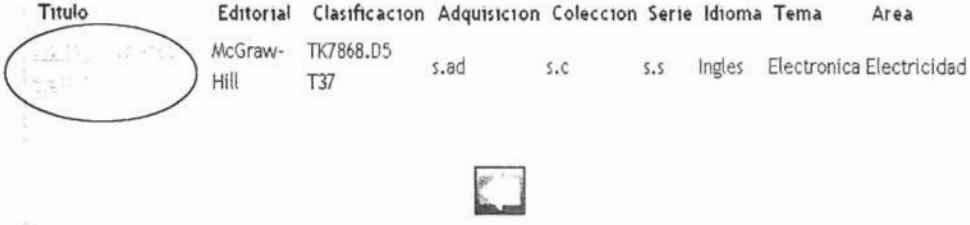
El usuario tendrá que dar clic para acceder al material relacionado con dicho autor

Muestra los autores con la letra seleccionada

El resultado será el siguiente:



Obteniendo así la información solicitada con sus características generales.



Ahora bien si el usuario requiere saber en que formato se encuentra solo basta dar clic sobre el título y lo llevara a dicha sección.

BUSQUEDA SENCILLA POR TITULO

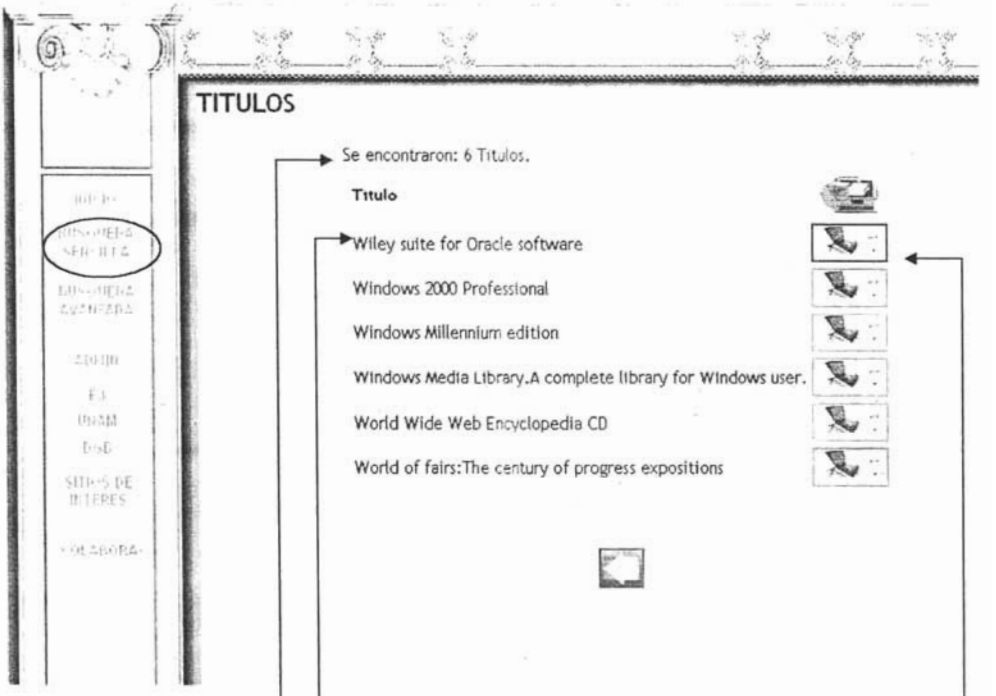
Seleccione la letra del comienzo del apellido del autor para ir directamente a dicha sección

A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - R - S - T - U - V - W - Y - Z

Seleccione la letra para ir directamente al titulo

A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - R - S - T - U - V - W - Y - Z

El mismo procedimiento se realiza para la búsqueda por titulo.

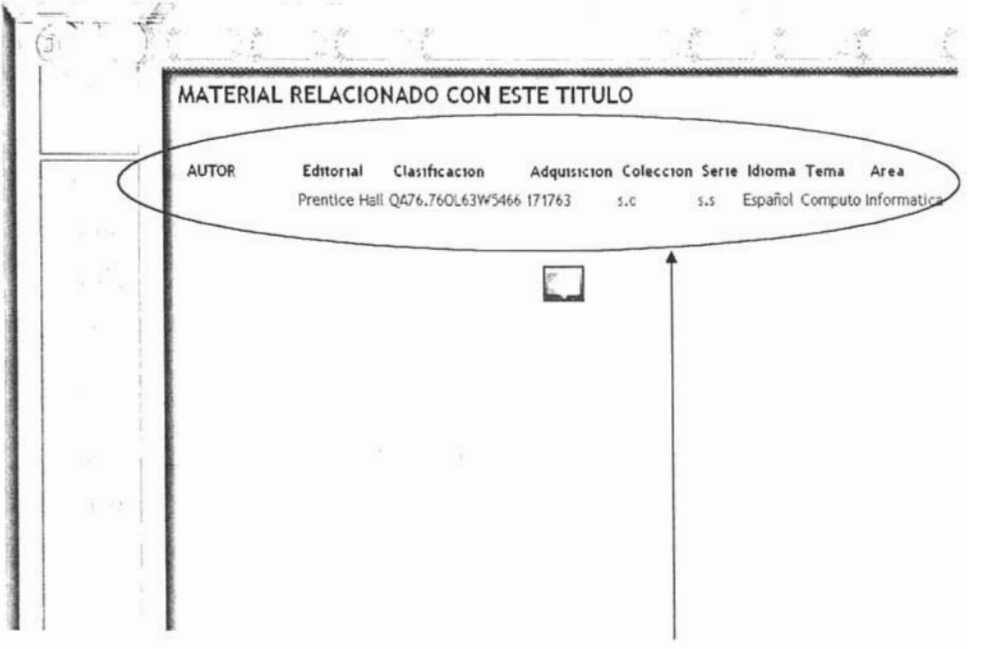


Muestra los títulos con la letra seleccionada.

Nos muestra el número de resultados encontrados sobre dicha petición.

El usuario tendrá que dar clic para acceder a la información acerca de ese título.

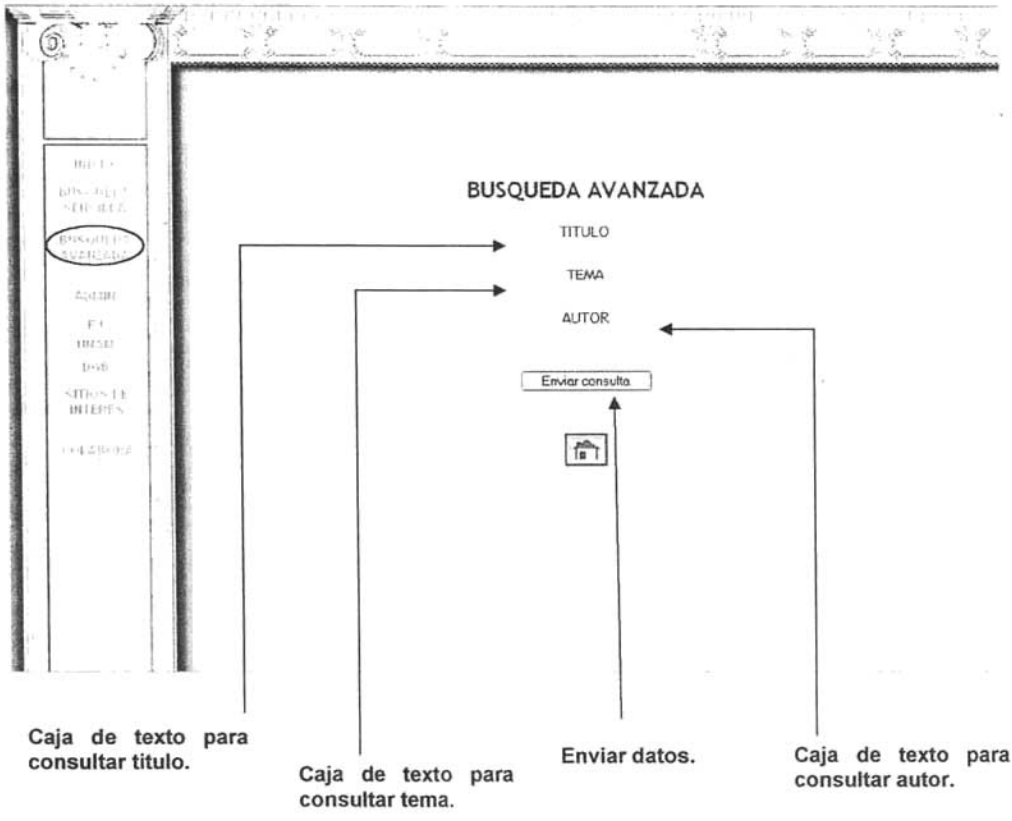
El resultado será el siguiente:



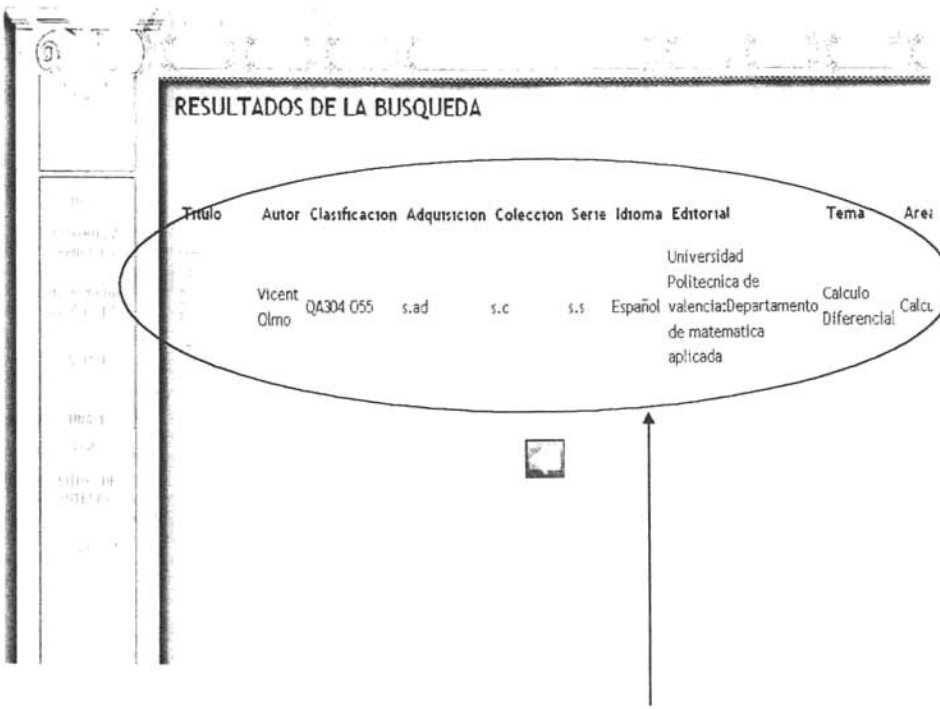
Obteniendo así la información solicitada con sus características generales.

Ahora si el usuario da un clic sobre el autor desplegará una pantalla con el tipo de formato sobre ese material.

BUSQUEDA AVANZADA.



Si el usuario requiere hacer una búsqueda más específica sobre el material que desea consultar ya sea por titulo, tema y autor el resultado será el siguiente:



Para ver las características del título basta con dar clic sobre el mismo, enseguida desplegará una pantalla con los resultados y el tipo de formato que se tiene acerca de ese material.

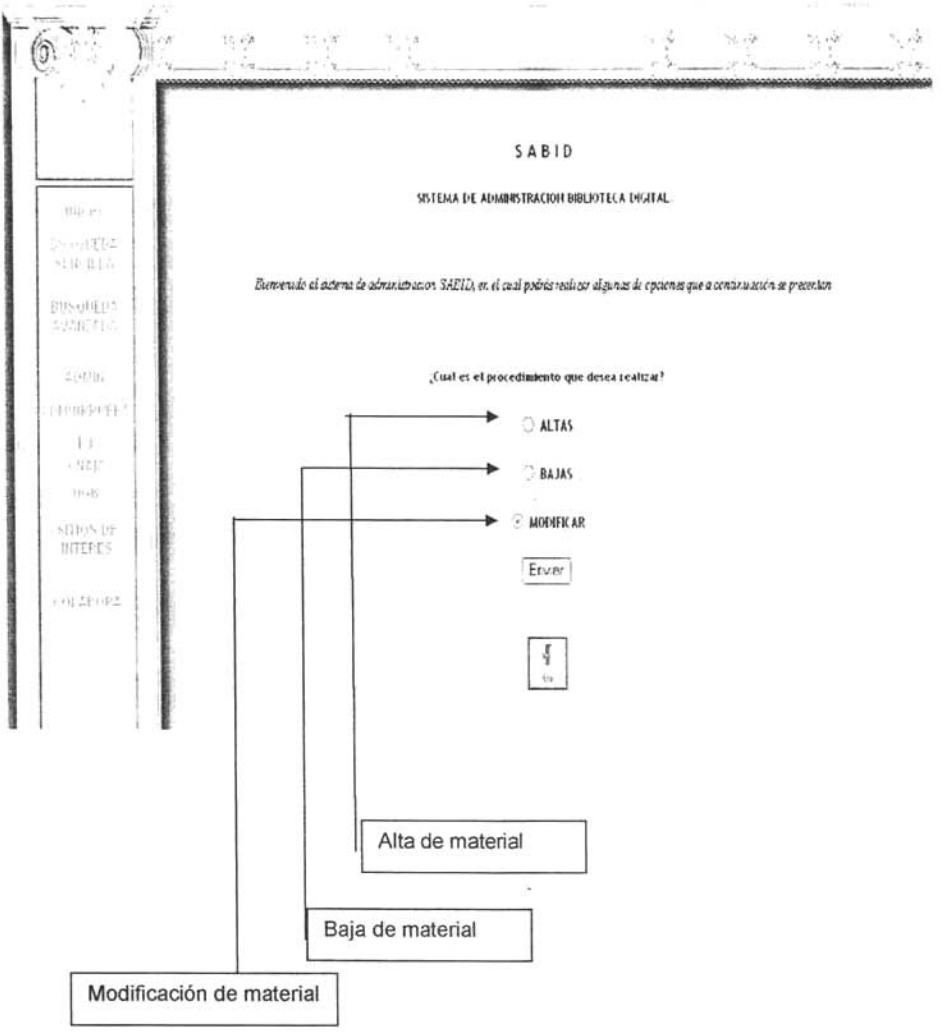
Ahora si el usuario desea realizar cualquiera de las tres opciones solo basta con llenar el campo indicado y realizar el mismo procedimiento.

ADMINISTRADOR.

Es una de las más importantes del sistema pues en ella solo tendrá acceso el administrador el cual se encargará entre otras cosas de dar de alta, cambios y baja del material. Como se explicó anteriormente se accede desde el menú principal. Al dar un clic nos mandará a una pantalla donde se pedirá un login y password.

Si los datos fueron introducidos correctamente entonces nos llevará a la página de bienvenida en caso contrario nos mostrará un mensaje de que los datos no fueron los correctos.

Una vez realizado lo anterior se mostrarán las opciones ya sea para realizar una alta, una baja o una modificación del material.



Ahora que se ha logrado entrar correctamente tenemos dos opciones si elegimos dar de alta, nos llevará a la siguiente pantalla. En ésta observamos todos y cada uno de los campos que deben de ser llenados para **alta del material**.

Mostramos en partes el contenido de la página pues por su contenido resulta extensa para ser mostrada completamente.

ALTA DEL MATERIAL

DATOS DE CONTRAPORTADA

Editorial	Ventana Press
Apellido Paterno	Vigueras
Apellido Materno	Paez
Nombre	Marco Antonio
Tema General	Redes
Area Especifica	Telecomunicaciones
Idioma	Español
Año de Publicacion	2000

En la pantalla anterior se deben llenar todos los campos de contraportada los datos generales del material. Posteriormente seleccionamos el tipo de soporte del mismo.

DATOS GENERALES DEL MATERIAL

Titulo Introduccion y Aplicaciones en Redes

Clasificacion TA345.128

Adquisicion 1602

Coleccion Computo **Serie** Aprendiendo Computo

Autor Coorporativo

ALTA DE SOPORTE

CD ☑

DISKETTE ☐

DVD ☐

VIDEO ☐

Una vez seleccionado el tipo de soporte se introducen las características en las cuales se puede leer la información del material en cuestión. Con ello mandamos la información.

Cuando se da de alta el material y los datos en los campos fueron llenados nos mostrará un mensaje diciendo que la inserción fue realizada con éxito.

ALTA DE MATERIAL

Cuando la inserción no se realizó con éxito nos mostrará la siguiente pantalla.

ACTUALIZACIÓN DE MATERIAL.

Ahora si se quiere **modificar** un material tenemos que seleccionar en la página del administrador. Nos mostrará la siguiente pantalla. Aquí solo basta con introducir el titulo y la clasificación del material y nos llevará a una pantalla donde se encuentran todos los datos de ese material y el administrador podrá realizar las modificaciones correspondientes.

ACTUALIZACION DE MATERIAL

DATOS PARA REALIZAR BUSQUEDA DEL MATERIAL A MODIFICAR

Titulo	Circuitos
Clasificacion	TDJ

Enviar datos | Borrar

Una vez realizado lo anterior se envían los datos y si existe algún error el sistema lo notificará.

BAJA DE MATERIAL.

El mismo procedimiento que la opción modificar se aplica en esta opción, sólo que aquí se van a eliminar todos los datos sobre el material.

Como en las opciones anteriores nos mostrará un mensaje si se realizó con éxito la eliminación o no.

CAPÍTULO 5

CONCLUSIONES

Con esta propuesta de aplicación la biblioteca del Anexo de la Facultad de Ingeniería será capaz de manejar su información digital de una manera más rápida y eficiente, pues aun no se cuenta con una herramienta que los auxilie en este tipo de tareas.

Además de brindar a los usuarios la oportunidad de conocer el acervo multimedia que en la mayoría de los casos se tiene poco conocimiento de existencia.

El diseño del sistema nos permite agregar funciones futuras a cada uno de los módulos. Esto tomando en cuenta que el sistema pasa por un ciclo y es en este punto en el que se pueden mejorar cada uno de ellos para un mayor beneficio de acuerdo a las necesidades que se presenten. Es importante señalar que los materiales digitales están cada vez abarcando campo en las bibliotecas y que representan una opción viable para los usuarios, no sólo por su fácil consulta sino por su cómoda adquisición.

El sistema cuenta con un entorno gráfico que servirá como herramienta para el administrador, que le permitirá realizar las tareas de alta, modificación y baja del material.

GLOSARIO DE TERMINOS Y ANEXOS.

ANSI: American National Standars Institute.

Base de Datos: Es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización.

Clave Libre: identificador único de cada uno de los materiales digitales existentes.

CSS (Cascade Style Sheets).

DTD: (Document Type Definition) documentos de definición de tipo.

DML: Lenguaje de Manipulación de Datos.

DBA: Administrador de las Bases de Datos (Data Base Administrator).

DDL: Lenguaje de Definición de Datos.

DCL: Lenguaje de Control de Datos.

DD: Diccionario de Datos.

DBMS: Sistema Manejador de Bases de Datos.

FK: Clave Foránea (Foreign Key) sirve para especificar la naturaleza de la relación entre las tablas, ya que hace referencia a una clave definida con anterioridad en cualquier otro lugar de la base de datos.

GNU: Es un acrónimo recursivo para ``GNU No es Unix''.

HTML: HyperText Markup Language (Lenguaje de Marca de Hipertexto)

Información digital: Es una forma de almacenamiento y de manipulación de colecciones de datos digitalizados.

Middleware: Es un término vago que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores.

Modelo de Datos: Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos.

Multimedia: Es una colección de tecnologías basadas en la utilización del ordenador que da al usuario la capacidad de acceder y procesar información en por los menos tres de las siguientes formas; texto, imagen (estática o en movimiento) y sonido.

PHP: Hypertext Preprocessor

PK: Clave Primaria (Primary Key) de la tabla, la cual hace que cada fila de dicha tabla sea exclusiva.

PL/SQL: (Procedural Language /SQL).

QBE: (Query-By-Example).

RDBMS: Sistema Manejador de Bases de Datos Relacional (Relational Database Management System).

Schemas: Describen la estructura de la información.

SGML: (Standard Generalized Markup Language) o Lenguaje de Marcas Generalizado.

Sistema Operativo: Es un programa que permite administrar los recursos de una computadora, como lo son: Memoria, CPU, dispositivos de E/S (Unidades de Discos, monitor, teclado, etc.).

SQL: Structured Query Language (Lenguaje de consultas estructurado).

SGBD: Sistema Gestor de Bases de Datos.

Tablas: Mecanismos de almacenamiento de datos dentro de una base de datos.

UML: Lenguaje de Modelado Unificado.

XML: Lenguaje de Marcas Generalizado (Extensible Markup Language).

XSL: Extensible Style-sheet Language.

XHTML: Es una familia de módulos y tipos de documentos que reproduce, engloba y extiende HTML.

BIBLIOGRAFIA.

http://www.desarrolloweb.com/manuales/6/

http://pisuerga.inf.ubu.es/lsi/Docencia/TFC/ITIG/icruzadn/Memoria/24.htm

http://www.une.edu.mx/cursos/vb50_1.htm

http://www.ctipmagazine.org.py/setiembre2003/uml.html

http://rinconprog.metropoliglobal.com/CursosProg/BDatos/index.php

http://www.php.net

Web Database Applications, Hug E.Williams Ed.Oreilly

Bases de datos y su aplicación con SQL / Sergio Ezequiel Rozic, Ciudad de Buenos Aires, Argentina: MP Ediciones, 2004

PHP and MySQL Web development / Luke Welling and Laura Thomson, Welling, Luke, 1972-, impreso Indianapolis, Indiana: Sams, 2001

PHP bible / Tim Converse and Joyce Park, 2nd ed.

Ingeniería de software / Ian Sommerville, México: Pearson Educación, c2002

Beginning PHP, Apache, Mysql Web Develoment, Michael K Glass, Yann Le Secovamec, Ed. Wiley Publishing, Inc.

Manual de Transact-SQL user guide Enero 2000, Dirección de Administración Escolar

Notas del Curso de Transact –SQL , Dirección General de Servicios de Cómputo Académico.

Manual de Programación, Stig Saether Bak Ken, Alexander Aulbach.