3

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE APLICACIONES A BAJO COSTO PARA INTERNET

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

GILBERTO ERIC APARICIO GUERRERO

DIRECTOR DE TESIS:

ING. ALBERTO GONZÁLEZ GUIZAR

MÉXICO, D.F. 2001.

Agradecimientos.

A mis padres:	*Por haberme inculcado el hábito del estudio y la superación. Gracias por apoyarme durante toda mi carrera.*
A mis hermanas Arcelia, Alma y Mayra:	*Por haberme asistido de distintas formas durante mi educación y tener paciencia de mi mal humor durante los tiempos que me encontré bajo mucha presión.*
A Fabiola:	*Por estar a mi lado alentándome a continuar hacia adelante y brindarme su amor.*
A Alberto:	*Por ser un buen amigo y ayudarme con este trabajo. Gracias por dirigir mi tesis.*
A mis amigos de Ingeniería:	*Que sin su amistad y apoyo hubiera sido más difícil salir adelante en los momentos en los que uno esta a punto de vencerse.*
A los instructores, coordinadores y compañeros de la Dirección de Cómputo para la Administración Académica:	*Por haberme permitido enriquecer mis conocimientos a partir de los suyos y brindarme su amistad que fue fundamental para un mejor desempeño académico y profesional.*
A los profesores que disfrutan su labor en la Facultad de Ingeniería:	*Que con su magnífico trabajo nos inspiran, nos alientan a seguir adelante a pesar de los tropiezos y son una guía para llegar a lograr nuestra meta: Ser buenos Ingenieros, orgullosamente de la UNAM.*

ÍNDICE

I. INTRODUCCIÓN

Actualmente existen diversas herramientas de desarrollo de aplicaciones para Internet, pero varias de ellas tienen la desventaja de estar demasiado ligadas a una tecnología específica o ser demasiado caras, debido a los requerimientos de software y/o de hardware.

El desarrollo de aplicaciones para Internet, podría realizarse con un bajo costo, si se eligieran las herramientas adecuadas de programación que permiten interactuar con el elemento fundamental en este tipo de aplicaciones: las bases de datos.

La tecnología avanza demasiado rápido y es cada vez más difícil de mantenerse a la vanguardia, sobre todo cuando el costo de los equipos y software es muy elevado y estos se vuelven obsoletos en poco tiempo. Por ello, es necesario evaluar herramientas que permitan mantener un bajo costo durante la etapa de desarrollo de prototipos y la creación de la aplicación misma.

Una vez terminada la aplicación se puede realizar la adquisición del equipo conveniente que cumpla con los requerimientos reales de servicio. Para que esto sea posible, es indispensable que el sistema operativo a elegir, funcione tanto en equipos de bajo costo que serán ocupados durante el desarrollo, como en servidores especializados de alto costo donde residirá la aplicación final. Otro punto importante es que las herramientas de desarrollo no deben de ser demasiado complejas, para que cualquier modificación necesaria a la aplicación desarrollada, requiera de poco esfuerzo.

I.1. Objetivo.

Tomando en cuenta los planteamientos anteriores, se elaborará una aplicación para el Banco de Información Veterinaria (BIVE) en Internet. El banco de información consiste en referencias de literatura no convencional como son: congresos, apuntes, artículos, etc. La aplicación desarrollada deberá manipular toda esta información de manera eficiente en su almacenamiento, modificación y sobre todo, en la consulta; todo ello a través de Internet.

I.2. Resultados esperados.

Se espera que la aplicación desarrollada para el Banco de Información Veterinaria, mantenga un bajo costo de desarrollo, sin que ello limite la flexibilidad, portabilidad, escalabilidad y que a su vez, sea de fácil manejo para los usuarios finales, los cuales no requiera de grandes conocimientos técnicos para actualizar, mantener y consultar la información disponible a través de un navegador de páginas de Internet.

Adicionalmente la aplicación debe ser, desde el punto de vista de programación, fácil de retomar y modificar para que en caso de requerirse alguna corrección o adición, cualquier programador (sin necesidad de tener amplio conocimiento y dominio del lenguaje), sea capaz de realizar el mantenimiento necesario.

El trabajo realizado puede promover el desarrollo de aplicaciones en general para Internet dado el bajo costo para su desarrollo, además de servir como modelo para la creación de nuevas aplicaciones, dentro y fuera de la UNAM.

II. ANTECEDENTES

II.1. Evolución de Internet.

Internet tuvo sus orígenes en un proyecto de tipo militar que comenzó a finales de los años sesenta. Su objetivo era mantener una comunicación entre una serie de computadoras, sin verse afectada en el caso de que alguna máquina de la red fuera dañada.

Posteriormente algunas universidades se conectaron a esta red llamada ARPANET (Advanced Research Project Agency Network; Red de la Agencia de Proyectos de Investigación Avanzada), la cual fue financiada por el Departamento de Defensa de los Estados Unidos durante varios años. A partir del contacto con las universidades, la cantidad de usuarios fue aumentando, sin embargo continuó siendo utilizada únicamente con fines académicos.

La tecnología de comunicaciones fue avanzando rápidamente, con lo que no sólo grandes instituciones tenían la capacidad de conectarse a la red, sino que también individuos y organizaciones mediante el empleo de módems. En un inicio la información disponible en Internet era básicamente de tipo académico, y la presentación de la información era en texto *ASCII* sin ningún formato.

La idea de Internet llamó la atención de la gente de negocios, y a principios de 1990, empresarios y otros usuarios ejercieron presión para que Internet abriera sus puertas a otras organizaciones, no necesariamente académicas.

Las necesidades para presentar la información a una comunidad variada de usuarios cambiaron. La restricción de presentar únicamente texto limitaba la capacidad de comunicación, algo que podía mejorarse mediante el empleo de imágenes.

Un gran impulso al crecimiento de Internet, se originó con el surgimiento de un lenguaje que permitía integrar Imágenes y texto en un documento: *HTML* (Hypertext Text Markup Language; Lenguaje de Marcado de Hipertexto). Además surgieron programas con una interfaz gráfica para poder visualizar los documentos de *HTML*, a partir de este momento el manejo de Internet se simplificó para el usuario, que ya no tenía que aprender a utilizar programas complejos y disponía de una presentación enriquecida con imágenes y ligas que relacionaban a un documento con otro (ligas de *hipertexto*).

Internet vino a cambiar la manera de comunicación y distribución de información. No solamente se vio beneficiado el campo comercial, sino el de investigación. Por ejemplo, si se habla de un proyecto de investigación, los resultados pueden ser publicados en Internet e inmediatamente están disponibles para millones de usuarios a través de la red, sin necesidad de esperar que se realice una impresión y envío por correo tradicional, lo cual tomaría varias semanas.

Internet sigue evolucionando, tanto en tecnología de hardware como de software. Continuamente nacen nuevas tecnologías que permiten enriquecer el contenido de Internet y varias de ellas están disponibles de manera gratuita.

II.2. Arquitectura Cliente/Servidor.

En un principio, las aplicaciones realizaban todos sus procesos en una sola máquina con gran poder de procesamiento, donde terminales sin capacidad de procesamiento (llamadas terminales tontas) eran conectadas al equipo central. Sin embargo esto traía como consecuencia que si se deseaba cambiar alguna parte del sistema, el resto de la aplicación se veía afectada también.

Conforme el poder de cómputo fue avanzando, se consideró distribuir la carga de procesamiento entre el equipo central y las computadoras conectadas a dicho equipo. De esta forma, surgió el concepto de la arquitectura Cliente/Servidor con dos, tres o más capas, donde cada capa tiene un cierto grado de independencia respecto a las demás capas, permitiendo realizar algunos cambios en la capa sin afectar a las demás. El cliente es un proceso que se encarga de realizar peticiones de servicios a un servidor, el cual determinará la forma de atender dicha petición.

Las capas lógicas que componen a una Arquitectura Cliente/Servidor son:

- Capa de presentación. Está capa se encarga de interactuar con el usuario que realiza una petición a un servidor. Presenta y recibe información del usuario, el cual no necesita saber los detalles de lo que existe detrás de esta capa. Un navegador de páginas de Web, es un ejemplo de software que abarca esta capa.

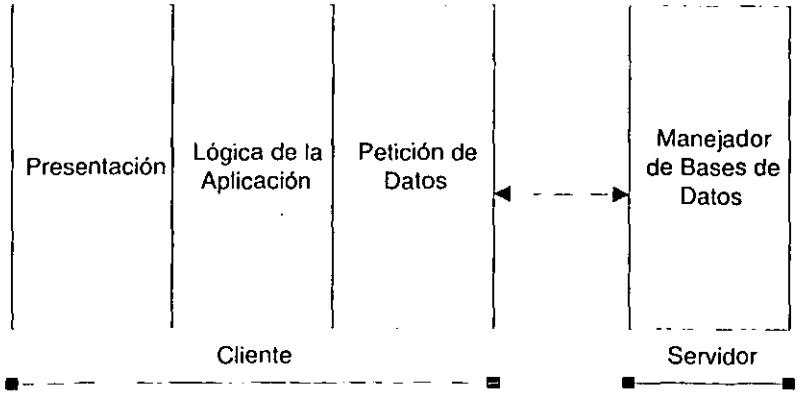
- Capa de la lógica de la aplicación. En esta capa se procesa tanto la información recibida del usuario como la que se le enviará. La capa de la lógica involucra la codificación de la aplicación en algún lenguaje de programación.

- Capa de petición de datos. Esta capa se encarga de realizar la petición de información requerida, a la capa de acceso a datos. Las peticiones se hacen a través de un lenguaje de consulta de datos donde se especifica que información se desea, sin necesidad de especificar los procedimientos a emplear para obtener dicha información de los dispositivos de almacenamiento. Un lenguaje ampliamente usado es *SQL* (Structured Query Language; Lenguaje Estructurado de Consulta).

- Capa de acceso a datos. La capa está relacionada con los procesos para almacenar y recuperar la información del dispositivo de almacenamiento. La consulta y almacenamiento de información se realiza de manera muy eficiente mediante el uso de los manejadores de bases de datos.

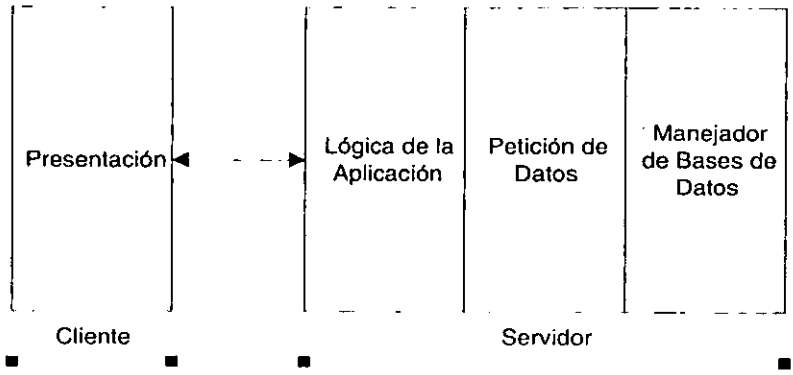
La carga de procesamiento se puede distribuir entre las distintas capas con que cuente la arquitectura, estas podrían estar físicamente distribuidas en distintos equipos de computo a través de la red, o podrían residir dentro de un mismo equipo.

Si las capas de presentación, lógica de la aplicación y petición de datos se encuentran físicamente del lado del cliente y del lado del servidor únicamente se tiene al manejador de bases de datos, la arquitectura se llama "Cliente/Servidor de dos capas con cliente grueso", ya que el procesamiento en su mayoría reside en el cliente y no en el servidor. Este tipo de arquitectura no es muy empleada en las aplicaciones de Web.



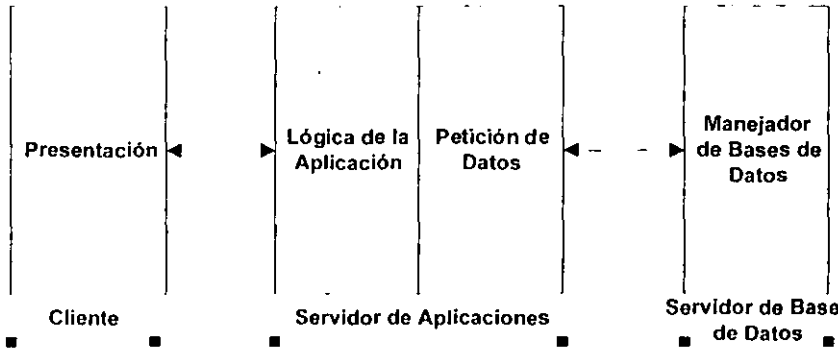
Diag. 1. Arquitectura Cliente/Servidor de dos capas con cliente grueso

Si la capa de presentación es la única que reside del lado del cliente y todas las demás se encuentran del lado del servidor, entonces la arquitectura es "Cliente/Servidor de dos capas con servidor de transacciones". Esta arquitectura se puede emplear para realizar aplicaciones de Web, donde la capa de presentación es el navegador de Web y las capas restantes residen en un servidor.



Diag. 2. Arquitectura Cliente/Servidor de dos capas con servidor de transacciones

Por último, cuando la capa de presentación reside en el cliente, las capas de la lógica de la aplicación y la de petición de datos residen en un equipo llamado "Servidor de Aplicaciones" y el manejador de bases de datos reside en otro equipo, se está hablando de una Arquitectura Cliente/Servidor de Tres capas. Esta capa es muy similar a la anterior, por lo que también puede ser empleada en una aplicación de Web.



Diag. 3. Arquitectura Cliente/Servidor de 3 capas

En todos los casos cuando se habla de una arquitectura de dos, tres o más capas, no se refiere al número de capas lógicas que componen la arquitectura, sino a la división física de éstas en dos, tres o más capas.

En la práctica las divisiones de las capas lógicas no siempre es tan marcada, ya que en ocasiones pueden combinarse. Por ejemplo, el cliente representado por el navegador de Web, puede realizar algunas validaciones que aseguren que la información a ser enviada al servidor cumple con ciertas reglas que impone la aplicación de Web, en este caso se distribuye la lógica de la aplicación entre el cliente y el servidor.

Dependiendo del tipo de aplicación, se puede disponer de distintas herramientas que permiten realizar un menor o mayor procesamiento del lado del cliente.

Los beneficios de las aplicaciones de Web, radican en que no es necesario diseñar, programar ni distribuir, la capa de presentación, ya que esta es implementada mediante un navegador de Web.

II.3. Software de distribución libre.

El software libre nace con la idea de compartir y mejorar software, para el bien de toda la comunidad de usuarios. El fundador del proyecto de software libre fue Richard Stallman.

El software libre se refiere a la libertad de los usuarios de correr, copiar, distribuir, estudiar, cambiar y mejorar el software. Existen cuatro libertades principales que deben tener los usuarios:

- La libertad de correr el programa, con cualquier propósito.
- La libertad de estudiar como funciona el programa, y adaptarlo a sus necesidades. El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias de manera que se puede ayudar al vecino.
- La libertad de mejorar el programa, y liberar las mejoras al publico de tal manera que toda la comunidad se beneficia. El acceso al código fuente es una condición previa para esto.

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

Las ventajas de desarrollar con programas de distribución libre, radican en que se pueden realizar los cambios necesarios al software sin necesidad de solicitar autorización para ello ni realizar pago alguno.

El software siempre es susceptible a errores de programación y cuando el software es *propietario* y tiene un error (*Bug*), el usuario tiene que esperar a que la compañía de software decida repararlo. En el software de distribución libre, los usuarios pueden esperar a que algún otro usuario distribuya la corrección, pero siempre se tiene la posibilidad de corregirlo por sí mismo.

La ventaja de tener el código fuente de la aplicación radica en que se fomenta la participación de múltiples usuarios, los cuales pueden participar en el desarrollo de dicha aplicación para llevarla a otras plataformas de hardware y software. Esto beneficia a todos los usuarios, ya que las aplicaciones no están limitadas a funcionar en una sola plataforma.

El empleo de software de distribución libre permite una disminución en los costos de desarrollo, dado que el software se puede adquirir por una pequeña cuota o sin cargo alguno cuando se consigue a través de Internet, además el software se puede modificar en caso de ser necesario. Desde el punto de vista de la implementación, no es necesario pagar licencias para cada máquina que utilice dicho software.

Una probable desventaja al utilizar programas de distribución libre, es que nadie está obligado a brindar soporte, sin embargo, este siempre existe ya sea a través de correo electrónico, listas de discusión, respuestas a preguntas frecuentes, manuales, tutoriales.

Los programas de distribución libre tienen una licencia *GPL* (General Public License, Licencia Pública General) en donde se expresa las libertades y obligaciones que tiene el usuario. La licencia *GPL* se puede encontrar en http://www.gnu.org

II.4. Ingeniería de Software.

La ingeniería de software se puede definir como: "La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir la aplicación de ingeniería al software".[1]

Las actividades asociadas a la ingeniería de software se pueden dividir en tres grandes rubros:

- Definición. Es la etapa en la cual se realiza una análisis sobre que es lo que se va a procesar, el funcionamiento, rendimiento y comportamiento que tendrá la aplicación a desarrollar.
- Desarrollo. Esta etapa consiste en traducir el análisis, realizado en la etapa anterior, en un diseño técnico y en código que brindará la funcionalidad requerida.
- Mantenimiento. En esta etapa se realizan las adaptaciones y correcciones necesarias a la aplicación desarrollada.

Para que un software pueda ser de calidad, las tres etapas deben ser planeadas cuidadosamente, basándose cada una de ellas en la anterior.

El enfoque de este trabajo, está orientado únicamente hacia el desarrollo de la aplicación, por lo que conviene recalcar que el lector no debe olvidarse que existe una etapa previa de la cual depende completamente el éxito de dicha aplicación.

[1] IEEE. Standars Collection: Software Engineering. IEEE Standard 610.12-1990.

III. DISEÑO DE APLICACIONES PARA INTERNET

III.1. Consideraciones previas al diseño de la aplicación.

El diseño de aplicaciones para Internet, involucra tener conocimiento indispensable de varios elementos que están involucrados el desarrollo de la aplicación:

- Es necesario conocer a fondo el sistema operativo en que se va a trabajar, no sólo al nivel de usuario sino al nivel de administración, ya que para instalar y configurar las herramientas a emplear para el desarrollo de la aplicación, es necesario contar con los conocimientos necesarios para poder llevar a cabo estas tareas de manera exitosa. La administración en los sistemas tipos *Unix*, no es sencilla, pero las ventajas que nos ofrece al estar disponible para gran variedad de plataformas y ser tan flexible cubren esa dificultad.

- Otro punto que es necesario considerar, es que se debe de tener conocimientos acerca de los manejadores de bases de datos, ya que estos juegan un papel muy importante dentro el desarrollo de aplicaciones en general. Los manejadores de bases de datos, permiten manipular la información de una manera eficiente, siempre y cuando se tenga un buen diseño de la base de datos. Los manejadores de bases de datos también involucran una administración que puede ser algo compleja, sobre todo si se desea optimizar al máximo el manejo de información.

- Contar con conocimientos de diseño de bases de datos, ya que esto es la clave para lograr el manejo eficiente de la información.

- Por último, es necesario entender el funcionamiento de las páginas de Internet, lo que implica un manejo del leguaje *HTML* y de los procesos que

involucra el manejo de información a través de la Web conociendo las limitaciones.

III.2. Elección de la plataforma de trabajo.

Durante la etapa de diseño es necesario elegir el tipo de sistema operativo, el servidor de Web, las herramientas de programación y el manejador de bases de datos. Esta elección no es sencilla, ya que cada una de ellas está relacionada con las demás, debido a que no todas las herramientas de programación están disponibles para *Unix* y NT, asimismo ocurre para los manejadores de bases de datos y servidores Web.

III.2.1. Sistema operativo.

El sistema operativo, abarca un conjunto de programas que permiten administrar los recursos de la computadora, además es la base para el funcionamiento de otros programas.

Para una aplicación de Internet es necesario disponer de un sistema operativo que brinde servicios a múltiples usuarios y que sea capaz de atender varias tareas al mismo tiempo. Este tipo de Sistemas operativos son los multiusuarios y multitareas.

Los principales tipos de sistema operativo multitarea y multiusuario son los de Windows NT de Microsoft y los de *Unix*, de los cuales existen diversas variantes.

Los sistemas operativos de tipo Windows NT, incluyen las versiones de Windows NT (Server y Workstation) y 2000 (Professional y Server).

Por otra parte los sistemas operativos de tipo *Unix* abarcan las siguientes variantes entre otras: Aix para equipo IBM; HP-UX para equipo Hewlet Packard; Solaris para equipo Sun Microsystems; Irix para equipo Silicon Graphics; FreeBSD, OpenBSD y *Linux* funcionan en diversos tipos de hardware; y Digital *Unix* para equipo de Compaq.

El sistema operativo se instala en el equipo que servirá como servidor, en donde se requiere atender a varios usuarios que consulten las páginas de Web a través de la red.

Las ventajas de elegir un sistema operativo de tipo *Unix*, radican en que existen versiones para distintas plataformas, desde computadoras personales hasta equipos de gran poder de computo como es el equipo mainframe. Por lo tanto, se puede estar seguro de que conforme vayan aumentando los requerimientos de procesamiento, se podrá disponer de equipo de gran poder de cómputo que funcione con el sistema operativo *Unix*. Es decir que utilizar sistemas operativos de tipo *Unix*, nos permite escalabilidad conforme se vaya requiriendo. El sistema operativo Windows NT no brinda tanta facilidad para realizar esta escalabilidad.

Otra ventaja de los sistemas operativos de tipo *Unix* es que existe una mayor cantidad de herramientas de distribución libre o que están disponibles gratuitamente, lo que permiten reducir los costos de desarrollo. Los sistemas operativos de tipo *Unix* en general requieren de una licencia comercial para poder utilizarlos, sin embargo hay algunos que son gratuitos para fines de desarrollo e incluso los hay de distribución libre como *Linux*, el cual no tiene restricciones del número máximo de usuarios conectados al sistema, debido a la naturaleza de su licencia *GPL*.

Para los sistemas operativos de tipo Windows NT no existen versiones que sean gratuitas y menos aún de distribución libre lo cual implica forzosamente un gasto para el pago de las licencias respectivas dependiendo del número de usuarios que trabajen con el sistema operativo. Además, no existe tanta disponibilidad de software de distribución libre o gratuito para esta plataforma.

Unix es extremadamente flexible, cosa que puede representar una ventaja o una desventaja dependiendo de los conocimientos y habilidades del administrador del servidor. La flexibilidad que brinda *Unix* permite una mejor integración entre el sistema operativo, el manejador de bases de datos y el servidor de páginas de Web, lo cual representa una mejora en el rendimiento del sistema.

En el caso de Windows NT, este no es tan flexible, sin embargo, las tareas de administración son más sencillas que en *Unix*, debido a que toda la administración se lleva a cabo mediante una interfaz gráfica

Unix brinda muchas ventajas y no es casualidad que varios de los sitios más importantes en Internet utilicen alguna variante de *Unix*.

En la siguiente lista se muestran el tipo de servidores que ocupan algunas de las empresas más importantes de Estados Unidos. En ella se puede observar una tendencia a predominar los sistemas operativos tipos *UNIX*.

Compañía u Organización	Tipo *Unix*	Tipo NT	Servidor de páginas de Web
ABC		☑	Microsoft IIS
Amazon.com	☑		Netscape Commerce
AOL	☑		NaviServer AOLserver
Apache	☑		Apache Web Server
AT&T	☑		Netscape Enterprise
Bank of America	☑		Netscape Enterprise

Compañía u Organización	Tipo *Unix*	Tipo NT	Servidor de páginas de Web
Barnes and Noble		☑	Microsoft IIS
Burger King		☑	Microsoft IIS
Chevrolet	☑		Netscape Enterprise
CNN	☑		Netscape Enterprise
The Coca-Cola Company	☑		Netscape Enterprise
Compaq		☑	Microsoft IIS
CompuServe		☑	Microsoft IIS
FBI	☑		Netscape Enterprise
Ferrari	☑		Netscape Enterprise
Ford		☑	Microsoft IIS
Harvard University	☑		Apache Web Server
Honda	☑		Netscape Enterprise
Informix	☑		Netscape Enterprise
Intel		☑	Microsoft IIS
Lamborghini	☑		Apache Web Server
Levi's		☑	Microsoft IIS
McDonald's	☑		Netscape Enterprise
Mercedes-Benz	☑		Netscape Enterprise
MIT	☑		Apache Web Server
Mitsubishi	☑		Netscape Enterprise
Motorola	☑		Apache Web Server
NASA	☑		Netscape Enterprise
Netscape	☑		Netscape Enterprise
Nissan	☑		MagnetWeb
Oracle	☑		Oracle Web Enterprise
PanAm		☑	Microsoft IIS
Panasonic	☑		Netscape Enterprise
Pepsi-Cola	☑		Netscape Enterprise
Philips	☑		Apache Web Server
Pizza Hut		☑	Microsoft IIS
RCA	☑		Netscape Enterprise
Sony	☑		Netscape Enterprise
Sybase	☑		Netscape Enterprise
Texaco		☑	Microsoft IIS
Texas Instruments	☑		Apache Web Server
Unisys		☑	Microsoft IIS

Compañía u Organización	Tipo *Unix*	Tipo NT	Servidor de páginas de Web
Volvo	☑		Apache Web Server
Wal-Mart	☑		Netscape Enterprise
Yahoo!	☑		Desconocido

En México las principales compañías e instituciones educativas siguen esta misma tendencia, aunque no tan marcada:

Compañía u Organización	Tipo *Unix*	Tipo NT	Servidor de páginas de Web
Aeroméxico	☑		Apache
Banamex	☑		Netscape Enterprise
Bancomer		☑	Microsoft IIS
Bimbo	☑		NCSA
Bolsa Mexicana de Valores	☑		Netscape Enterprise
Cruz Azul		☑	Microsoft IIS
Grupo Vitro	☑		
Mexicana de Aviación		☑	Microsoft IIS
Sabritas		☑	Microsoft IIS
Televisa		☑	Microsoft IIS
Telmex	☑		Netscape Enterprise
Tv Azteca	☑		Netscape Enterprise

Compañía u Organización	Tipo *Unix*	Tipo NT	Servidor de páginas de Web
Instituto Politécnico Nacional	☑		Apache Server
Instituto Tecnológico Autónomo de México	☑		Apache Server
Tecnológico de Monterrey	☑		Apache Server
Universidad Autónoma de México	☑		Apache Server
Universidad Nacional Autónoma de México	☑		Apache Server

En la dirección: http://www.netcraft.com/whats/ es posible verificar esta lista. En este sitio, es posible introducir una dirección *HTTP*, mostrándose los detalles del sistema operativo y el servidor de Web empleados.

Desde el punto de vista de desarrollo, la ventaja es que se puede utilizar algún sistema operativo *Unix* de distribución libre o gratuito y así tener la seguridad de que la aplicación se puede llevar a muy diversos tipos de computadoras que pueden ser desde computadoras portátiles hasta costosos equipos de gran capacidad de procesamiento.

Una de las múltiples variantes de *Unix*, es *Linux*, por lo cual estos dos sistemas operativos son muy parecidos. *Linux* puede ser distribuido libremente, es software libre en la versión actual y seguirá siéndolo en el futuro, debido a la naturaleza de la licencia *GPL*.

Algo a tener en cuenta es que *Linux* está desarrollado siguiendo un modelo abierto y distribuido, en lugar de uno cerrado y centralizado como la mayor parte del software. Esto significa que la versión actualmente en desarrollo es siempre pública (con un retraso de una o dos semanas) para que cualquiera pueda usarla. El resultado es que en cualquier momento que se añada una nueva funcionalidad y salga a la luz la nueva versión, ésta casi siempre tendrá errores, pero serán detectados y corregidos rápidamente, a menudo en cuestión de horas, ya que mucha gente trabaja en ello.

En contraste, el modelo centralizado y cerrado significa que hay sólo una persona o un equipo trabajando en el proyecto, y sólo publican software que ellos piensan que esté trabajando bien. A menudo esto conlleva largos periodos de tiempo entre versiones, largas esperas para la corrección de errores y un desarrollo más lento. Por supuesto que la última versión de este tipo de

software es a menudo de mejor calidad para el público, pero la velocidad de desarrollo es normalmente mucho más lenta.

Una ventaja de usar un sistema operativo *Linux* con fines de desarrollo es que es posible utilizarlo en computadoras personales que pueden ser desde 386 hasta las actuales Pentium. Esto reduce el costo del equipo a emplear durante el desarrollo.

En general los programas que estén disponibles para *Unix*, lo estarán de igual manera para *Linux*. Esto permite desarrollar a bajo costo utilizando software libre y equipo barato, y posteriormente tener la facilidad de migrar la aplicación a un equipo más potente que maneje *Linux* u otra variante de *Unix*.

III.2.2. Sistemas manejadores de bases de datos. DBMS (Data Base Management System)

Los *DMBS* (Database Management System; Sistema manejador de bases de datos) son programas que organizan la información de manera que su consulta, almacenamiento y modificación se lleve a cabo de la manera más eficiente posible. Actualmente los dos principales tipos de *DMBS* en el mercado son: los relacionales conocidos como *RDMBS* (Relational Database Management System; Sistema Manejador de Bases de Datos Relacional) y los orientados a objetos. De estos dos, las bases de datos relacionales se encuentra más difundidas, que las relativamente nuevas, bases de datos orientadas a objetos.

Los estándares están totalmente definidos para las bases de datos relacionales, por lo que su uso significa disponer de una mayor variedad de *DMBS* de donde elegir y de una mayor cantidad de bibliografía.

Los componentes principales de un *RDMBS* son los siguientes:

- DDL (Data Definition Language; Lenguaje de Definición de Datos). Este componente permite definir o describir los objetos de la base de datos. Puede usarse para crear, alterar o borrar relaciones (tablas), vistas, restricciones de integridad, etc.
- DML (Data Manipulation Language; Lenguaje de Manipulación de Datos). Apoya el manejo o procesamiento de la base de datos. Puede usarse para leer (consultar), modificar, borrar o agregar registros a las relaciones existentes. Una de las primeras funciones de los *DMBS* es la de soportar un DML en el cual el usuario pueda formular comandos que permitan manipular los datos.
- DD (Data Dictionary; Diccionario de datos). Es una base de datos que contiene información acerca de los objetos existentes en la misma base de datos.
- DCL (Data Control Language; Lenguaje de Control de Datos). Permite establecer los privilegios de acceso a los datos; en otras palabras, establece la seguridad de la base de datos.

Las diferencias entre un *RDMBS* y otro, radican principalmente en la forma de administrar la base de datos, dado que cada compañía implementa características adicionales que extienden la funcionalidad del *RDMBS* y aunque ambos tengan un DDL, este puede ser más flexible en uno de ellos. Por ejemplo, MySQL permiten agregar, borrar y modificar la definición de una columna dentro de una tabla existente en la base de datos, mientras que Adaptive Server de Sybase sólo permite agregar columnas, en caso de querer eliminar o modificar una columna, es necesario borrar la tabla y volverla a crear con la nueva definición.

Todo *DMBS* que proclame ser relacional debe manejar *SQL* (Structured Query Language; Lenguaje Estructurado de Consulta) como lenguaje para consultar y manipular la información de la base de datos. El desarrollo de *SQL* se remonta a principio de los años 70, en los laboratorios de IBM, donde se desarrollo un prototipo llamado "System R" el cual fue sometido a diversas pruebas, cuyos resultados alentaron el desarrollo de productos comerciales.

La compañía Oracle fue la primera en emplear *SQL* comercialmente en 1979, posteriormente se expandió su uso, dado la gran facilidad para aprenderlo ya que usa frases en Inglés y finalmente se adoptó como un estándar, respaldado por el *ANSI* (American National Standars Intitute; Instituto Nacional Americano de Normas) y la ISO (International Standars Organization; Organización Internacional de Normas), el cual aún está vigente, conocido como *ANSI SQL* 89.

En la siguiente tabla se muestran algunos de los *RDMBS* disponibles en el mercado:

Nombre	**Plataformas para las que está disponible**	**Licencias**
Oracle	Solaris	Gratuito con fines de desarrollo
	Linux	Gratuito con fines de desarrollo
	Windows NT	Gratuito con fines de desarrollo
Adaptive Server (Sybase)	Solaris	Necesario contar con licencia comercial
	Linux	Gratuito para fines de desarrollo
	Windows NT	Necesario contar con licencia comercial
MySQL	*Unix*	Distribución libre
	Windows	Necesario contar con licencia comercial
MS SQL Server	Windows	Necesario contar con licencia comercial
PostgreSQL	*Unix*	Distribución libre

Para realizar una aplicación se podría utilizar cualquiera de ellos y si posteriormente se requiere cambiar de *RDMBS*, las modificaciones necesarias para adaptarse al nuevo *RDMBS* serían mínimas, ya que cumplen con un estándar de *SQL* propuesto por la *ANSI* (American National Standars Institute; Instituto Nacional Americano de Estándares).

Todos los *RDMBS* mencionados anteriormente siguen el estándar *ANSI* SQL92, cumpliéndolo total o parcialmente. Por ejemplo, los *RDMBS* comerciales como Oracle 8, Sybase 11 y MS SQL Server 6 cumplen con el estándar completamente, mientras que los de distribución libre como MySQL 3.27 y PostgreSQL 7.0 omiten algunas funcionalidades.

Actualmente los *RDMBS* tratar de cumplir con un nuevo estándar el *ANSI* SQL99. Tanto Sybase como Oracle y Microsoft SQL Server ya soportan SQL99.

Para poder elegir un *RDMBS* es necesario conocer el tipo y volumen de información que se manejará. Posteriormente será necesario evaluar cuales *RDMBS* son aptos para el manejo de dicha información. Por ejemplo, el manejo de información multimedia, como videos, imágenes, documentos, etc., reduce la elección del *RDMBS* a los comerciales únicamente, dado que los de distribución libre no abarcan estos aspectos; en cuyo caso sería conveniente elegir el *RDMBS* que este disponible gratuitamente con fines de desarrollo, para poder mantener un costo bajo en esa etapa.

III.2.3. Servidores Web.

El servidor de Web, es el otro elemento importante que permite interactuar al cliente con el servidor. *HTTP* es el protocolo de comunicación que utilizan para transferir páginas de Web y sus componentes. El cliente realiza las peticiones al servidor y el servidor procesa la petición y regresa una respuesta. Aquí se debe retomar la arquitectura Cliente/Servidor que tienen las aplicaciones en Internet.

El servidor de Web es un programa, encargado de enviar el código *HTML* y otros componentes al navegador de Web, el cual lo interpretará y mostrará el contenido con su respectivo formato.

Existen distintos servidores de Web, que pueden ser comerciales o de distribución libre. En la siguiente tabla se muestra una lista de algunos de los servidores Web disponibles:

Servidor de Web	Sistemas operativos en los que se encuentra disponible	Costo aproximado en dólares
AOLserver	Digital *UNIX*, SCO, HPUX, Windows NT, *Linux*, Windows 95, FreeBSD, Windows 98, IRIX, Solaris	Gratuito. Software con licencia *GPL*
Apache	NetBSD, Digital *UNIX*, BSDI, AIX, OS/2, SCO, HPUX, Novell NetWare, Macintosh, Be OS, Windows NT, *Linux*, Windows 95, FreeBSD, Windows 98, IRIX, Solaris	Gratuito. Software con licencia *GPL*
Commerce Server/400	AS/400	$4,995
Enterprise Server	Novell NetWare	$1,295
Internet Information Server	Windows NT	Gratuito, incluido con el "NT 4.0 option pack"
Java Server	OS/2, HPUX, Windows NT, *Linux*, Windows 95, IRIX, Solaris	$295
Lotus Domino Go Webserver	Digital *UNIX*, AIX, OS/2, HPUX, Windows NT, Windows 95, IRIX, Solaris	$495

Servidor de Web	Sistemas operativos en los que se encuentra disponible	Costo aproximado en dólares
Netscape Enterprise Server	Digital *UNIX*, AIX, HPUX, Windows NT, IRIX	$1,295
Oracle Web Application Server	HPUX, Windows NT, Windows 95, Solaris	No especificado. Existe una versión gratuita para fines de desarrollo
Roxen WebServer	Digital *UNIX*, AIX, HPUX, Windows NT, *Linux*, Windows 95, FreeBSD, Windows 98, IRIX, Solaris	Gratuito. Software bajo licencia *GPL*
Zeus Web Application Server	NetBSD, Digital UNIX, BSDI, AIX, SCO, HPUX, *Linux*, FreeBSD, IRIX, Solaris	$1,699

Todos los servidores Web manejan el protocolo *HTTP*, aunque cada uno de ellos implementa ciertas características adicionales que distinguen a uno de otro, por ejemplo, cifrado de información mediante SSL (Secure Socket Layer; Capa de sockets seguros), soporte de tecnología *Java*, utlización de memoria para almacenar las páginas que son solicitadas con mayor frecuencia para optimizar el tiempo de respuesta, etc.

Para elegir un servidor de Web, es necesario observar en que plataformas funciona, cual es su rendimiento, las características adicionales que brinda y precio que se está dispuesto a pagar.

La elección de una herramienta de programación va totalmente ligada a la elección del servidor de Web, debido a que lo más conveniente es que exista una total integración entre ellos, evitando así algunos inconvenientes de seguridad que podrían presentarse.

Como se desea un desarrollo que implique bajo costo, es necesario elegir un servidor de Web de distribución libre, como el Apache Server o Roxen Web Server.

Apache Server, es ampliamente utilizado en la actualidad, mientras que Roxen Web Server apenas se empieza a difundir por lo cual no es tan conocido, sin embargo, brinda grandes facilidades para desarrollar aplicaciones para Internet. Ambos servidores son de distribución libre y funcionan bajo plataformas *Unix* y Windows.

Tanto Apache Server como Roxen Web Server, son muy flexibles y permiten utilizar ciertos lenguajes de programación para generar un contenido dinámico en las páginas de Internet y para tener la facilidad de realizar conexiones a bases de datos para manejar información e incluirla dentro del contenido de las páginas de Web.

El ser flexible implica, que se pueden integrar distintas tecnologías utilizando el mismo servidor de Web, por ejemplo, Roxen Web Server permite trabajar con *Servlets*, *JSP* (*Java* Server Pages, Páginas *Java* en Servidor), *PHP* (Hypertext Preprocessor; Preprocesador de *hipertexto*), Perl, *Rxml* (Roxen Markup Language; Lenguaje de Marcado de Roxen), etc.

Apache Server.

Apache Server es el servidor de Web más utilizado. Apache Server es un servidor de distribución libre, que surgió basándose en un servidor desarrollado en la NCSA (National Center for Supercomputer Applications, Centro Nacional para Aplicaciones de Supercomputadoras). La primera versión surgió en 1995 y poco a poco fue volviéndose popular. Actualmente se estima que el 60% de los servidores Web utilizan Apache.

Alrededor de Apache existen varios proyectos que buscan ampliar la funcionalidad del servidor, integrando distintas tecnologías como *Servlets*, *JSP*,

PHP, CGI, etc. Estas tecnologías permiten generar contenido dinámico en las páginas de Web.

Apache Server permite integrar tecnologías de cifrado de datos mediante SSL, lo cual es fundamental en aplicaciones que involucran transacciones monetarias o de información muy sensible.

Apache Server está disponible para múltiples plataformas de *Unix* y para NT. Esto asegura que Apache pueda ser empleado tanto en pequeños servidores, como en potentes equipos de cómputo.

Roxen Web Server.

Roxen cumple con las últimas versiones los estándares Web abiertos como son *XML*, *Java*, WAP (Wireless Application Protocol; Protocolo para Aplicaciones Inalámbricas) y SSL. Los beneficios de adoptar estándares abiertos son múltiples, ya que se evitan los efectos de estar ligados a una tecnología propietaria. Al usar los estándares, es mucho más sencillo encontrar personal para trabajar en los proyectos, además de tener una mayor flexibilidad al tener la facilidad de trabajar con múltiples tecnologías y no con una sola. Todas aquellas tecnologías que no son estándares tienden a hacerse obsoletas rápidamente.

Una de las grandes ventajas de Roxen Web Server radica en que puede ser empleado en múltiples plataformas de sistema operativo: tanto *Unix*, como Windows NT. Los programadores no requieren cambiar el código para llevarlo de una plataforma a otra. Es la misma filosofía que sigue *Java*.

En ocasiones la administración del servidor de Web, puede ser una tarea complicada, al tener que modificar archivos de texto para modificar la configuración. Roxen soluciona este problema, mediante una interfaz gráfica a través de una página de Web, que permite administrar los sitios de una manera sencilla.

Roxen Web Server, también permite el manejo de protocolos para dispositivos inalámbricos, como los actuales teléfonos celulares que permite conexión a Internet permitiendo visualizar páginas Web.

Permitir el manejo de múltiples manejadores de bases de datos, brinda una gran flexibilidad. Las bases de datos pueden ser empleadas para mantener el control de acceso al servidor, o para desarrollar una aplicación. La ventaja de tener una amplia variedad, es que durante el desarrollo se puede emplear un manejador de bases de datos muy barato o de distribución libre y posteriormente cambiar a otro mucho más potente que cumpla con los requerimientos de manejo de información.

Roxen Web Server también maneja cifrado de datos siguiendo el estándar SSL para establecer conexiones seguras. Esto se puede emplear para proteger la información que es enviada a través de Internet. En aplicaciones de comercio electrónico, es fundamental contar con un servidor de Web que brinde soporte SSL, para evitar que información confidencial viaje a través de Internet sin ser cifrada y de esta manera, prevenir fraudes.

Roxen Web Server almacena en memoria el contenido que es continuamente requerido, de manera que reduce el tiempo de respuesta, ya que no requiere leer nuevamente el contenido del disco duro.

III.3 Elección de la arquitectura Cliente/Servidor.

Dentro del diseño de una aplicación se tiene que decidir el esquema de la arquitectura Cliente/Servidor a emplear. Una aplicación para Internet puede ocupar un navegador de Web para abarcar la capa de presentación, logrando con ello, evitar la programación de una capa adicional que aumentaría el tiempo y costo de desarrollo. La ventaja para el usuario de utilizar un navegador de Web, es que la interfaz que éste brinda para visualizar páginas de Internet, será la misma que se emplea en la aplicación, lo que le evita la molestia de acostumbrarse a distintas interfaces de programas.

Dependiendo de la naturaleza de la aplicación, se tiene que considerar la forma de distribuir la carga de procesamiento entre el cliente y el servidor. En el capítulo II.2 se exponen las formas como se puede distribuir el procesamiento. La lógica de la aplicación puede estar contenida totalmente del lado del cliente utilizando herramientas de programación como *Java*, para programar *Applets* y el manejador de bases de datos se encuentra en el servidor. La desventaja de utilizar este modelo radica en que es mayor tiempo de espera para descargar la página por primera vez y sobre todo, se depende demasiado de la versión de *Java* incluida en el navegador de Web.

Otro modelo es tener la lógica de la aplicación distribuida entre el cliente y el servidor. Del lado del cliente se puede utilizar algún lenguaje *Intérprete* que venga incluido con el navegador, con la finalidad de revisar que la información que se dispone a enviar al servidor es válida y de esta manera liberar un poco la carga de trabajo del servidor. El servidor por su parte recibirá los datos provenientes del cliente y mantendrá la mayor parte de la lógica de la aplicación, además de tener la tarea de realizar la comunicación con la base de datos empleada.

Por último la capa del manejador de bases de datos, puede residir en el mismo servidor junto con la capa de la lógica de la aplicación, en cuyo caso sería una arquitectura de dos capas o puede estar en otro servidor, siendo una arquitectura de 3 capas.

La siguiente etapa de diseño involucra la elección de herramientas para cubrir cada una de las capas lógicas de la arquitectura Cliente/Servidor. La capa de presentación ya está cubierta por el navegador de Web, por lo que se deben escoger únicamente para las demás capas.

Entre el cliente y el servidor debe de existir una comunicación mediante un protocolo. *HTTP* es el protocolo estándar para manejo de páginas de Internet, este es soportado por los navegadores de Web que conforman al cliente, pero del lado del servidor debe de haber un software que entienda ese mismo protocolo y atienda las peticiones de esos clientes. Este software es el servidor de Web.

Existen múltiples lenguajes para desarrollar aplicaciones para Internet, por lo que no es una tarea fácil la de decidir cual elegir. En capítulos posteriores se expondrán las ventajas y desventajas de algunas de ellas.

IV. HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES PARA INTERNET

IV.1. HTML y XML.

"*HTML* es un lenguaje de composición de documentos y especificaciones de ligas de *hipertexto* que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque sí le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. *HTML* también le indica cómo hacer un documento interactivo a través de ligas especiales de *hipertexto*, las cuales conectan diferentes documentos..."[2]

Las instrucciones de *HTML* son indicadas a través de etiquetas que se encierran entre paréntesis triangulares, por ejemplo, Tesis. La etiqueta le indica al navegador de páginas de Web que muestre el texto "Tesis" de manera enfatizada. La etiqueta indica el comienzo de la instrucción y indica donde termina. Para la mayoría de las etiquetas de *HTML* se debe especificar una etiqueta de inicio <...> y otra de fin </...>

Todos los documentos *HTML* tienen un encabezado, y una sección de contenido, llamada cuerpo.

[2] HTML. La guía completa.

A continuación se muestra la estructura básica de un documento de *HTML*:

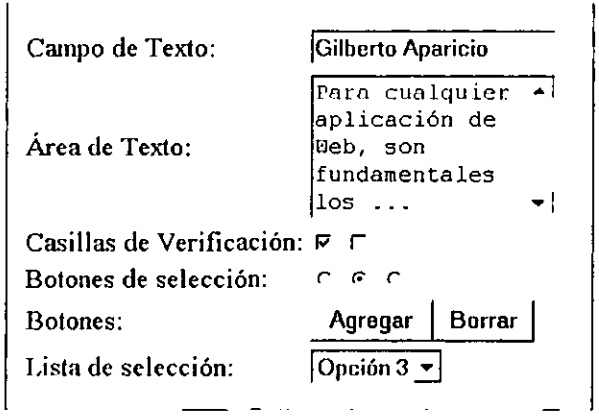
```
<HTML>
<HEAD>
  <TITLE>Título de la página</TITLE>
</HEAD>

<BODY>
En esta sección va el contenido de la página Web.
.
.
.
</BODY>
</HTML>
```

HTML es la parte fundamental de las páginas de Web y cualquiera que desee desarrollar una aplicación debe estar familiarizado con este tipo de código. *HTML* es un lenguaje muy sencillo que tiene demasiadas limitaciones, es por ello que es necesario complementarlo con algún lenguaje que permita ofrecer una mayor funcionalidad.

Para implementar una mayor funcionalidad del lado del cliente se emplean lenguajes que vienen incluidos con el navegador, como *Applets* de *Java*, *JavaScript* o VisualScript. Y del lado del servidor se utilizan diversos lenguajes que permiten generar contenido dinámico que puede provenir de una base de datos.

Para cualquier aplicación de Web, son fundamentales los elementos que permiten introducir información, para ser enviada al servidor. *HTML* maneja una etiqueta especial llamada <form>, para generar una forma que puede contener los siguientes elementos:



Todos estos elementos permiten introducir datos, seleccionar opciones e iniciar alguna acción a través de botones.

XML.

XML (Extensible Markup Language; Lenguaje extendible de Marcado) es una especificación desarrollada por la W3C (World Wide Web Consortium; Consorcio de la WWW). *XML* es una versión recortada de SGML (Standard Generalized Markup Language; Estándar de lenguaje de marcado generalizado), diseñado especialmente para los documentos de Web. Este permite a los diseñadores crear sus propias etiquetas personalizadas, habilitando su definición, transmisión, validación e interpretación de la información entre aplicaciones y entre corporaciones.

El que *XML* eventualmente sustituya *HTML* como el estándar para dar formato a los documentos de Web, depende demasiado en que sea soportado por las futuras versiones de los navegadores de Web. Por ejemplo, Microsoft Internet Explorer versión 5 maneja *XML*, pero en realidad las procesa como CSS (Cascade Style Sheets, Hojas de estilo en cascada), y Mozilla (Netscape) está aun experimentando con el soporte para *XML*. Por esta razón, aun no es conveniente utilizar *XML* para desarrollar aplicaciones de Web, cuya finalidad

es que estén disponibles para una gran variedad de usuarios, sin que estos tengan que preocuparse si su navegador soporta *XML*.

XML promete terminar con algunas limitaciones que tiene *HTML*, pero aun tiene que madurar, y este proceso tomará todavía algunos años.

Un ejemplo de aplicación *XML* es *XHTML* (Extensible Hypertext Markup Language; Lenguaje extendible de marcado de texto) el cual es un híbrido entre *HTML* y *XML* específicamente diseñado para dispositivos de red. *XHTML* básicamente facilita la transición de *HTML* a *XML*.

La versión actual de *XHTML* es la 1.0, y no agrega ninguna funcionalidad extra al *HTML* versión 4, sin embargo define un esquema que en un futuro podrá extender la funcionalidad de las páginas, mediante nuevas etiquetas y atributos que podrán ser agregadas a los ya existentes.

Las diferencias básicas de *XHTML* respecto a *HTML* radican en los siguientes puntos:

- Las etiqueta <head> y <body> no pueden ser omitidas.
- Dado que *XML* distingue entre mayúsculas y minúsculas, las etiquetas y atributos deben ser escritas en minúsculas.

HTML	`<INPUT type=TEXT VALUE="Escriba aquí">`
XHTML	`<input type="TEXT" value="Escriba aquí">`

- Los elementos no se pueden traslapar. Aunque en *HTML* el traslape no es permitido, la mayoría de los navegadores de páginas toleran este error, infiriendo cual es la sintaxis adecuada. *XML* más estricto en cuanto a sintaxis y no realiza esta corrección.

HTML	`<B><H1>Mensaje</B></H1>`
XHTML	`<b><h1>Mensaje</h1></b>`

- Todos los elementos no vacíos deben cerrarse con su elemento correspondiente. Nuevamente, los navegadores actuales, no verifican la terminación de algunas etiquetas, el ejemplo más conocido es el de la etiqueta <p> que indica un párrafo.

HTML	`Primer párrafo<p>` `Segundo párrafo<p>`
XHTML	`<p>Primer párrafo</p>` `<p>Segundo párrafo</p>`

- En los elementos vacíos se debe indicar la terminación. Los elementos vacíos son aquellos que no tienen una etiqueta de cierre, por ejemplo: <img>,
,<hr>, etc. es ellos se debe ocupar una diagonal para indicar la terminación del elemento

HTML	`El salto de línea en HTML es mediante  `
XHTML	`El salto de línea en XHTML es mediante  `

- El valor de los atributos debe ser entrecomillado aun cuando el valor sea numérico.

HTML	`<input type=text name="monto" value=4500>`
XHTML	`<input type="text" name="monto" value="4500" />`

- El valor de los atributos no puede ser minimizado. Un atributo es minimizado cuando sólo puede aceptar un valor.

HTML	`<option selected value="MX">Méx.`
XHTML	`<option selected="selected" value="MX">Méx.</option>`

IV.2. Lenguajes de Programación.

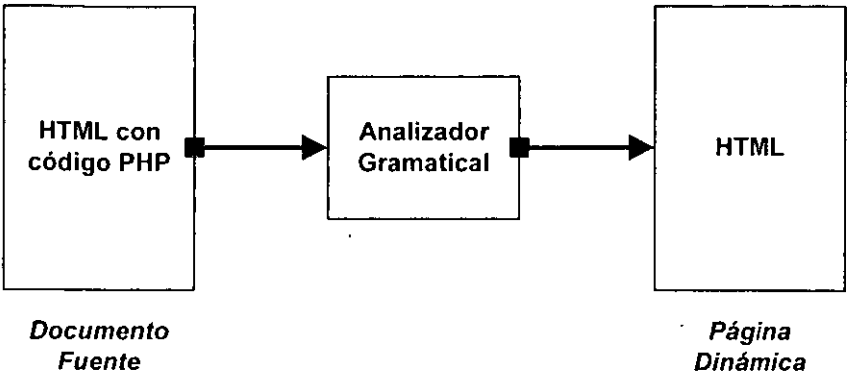
IV.2.1. PHP (Hypertext Preprocessor; Preprocesador de hipertexto)

PHP es un lenguaje de tipo script que se ejecuta en el servidor de Web. Es una herramienta que permite crear páginas dinámicas y hace posible que las paginas Web sean tratadas como una página *HTML* común, de igual forma su creación y edición es igual a la de cualquier documento *HTML*, a diferencia de scripts CGI, es que en lugar de escribir programas con instrucciones que reflejaran el resultado como una salida *HTML*; con *PHP* se escribe un script incrustado en el documento *HTML*, cuyo resultado será visualizado directamente el desplegarse dicho documento. El código *PHP* esta contenido en etiquetas especiales que indican el comienzo y fin del código, permitiendo conmutar entre el código *PHP* y *HTML*.

```
<HTML>
<BODY>
<?php
//Esto es un código PHP
print "Bienvenido $usuario";
?>
</BODY>
</HTML>
```

Lo que distingue a *PHP* de scripts como *JavaScript* es que el código es ejecutado en el servidor con lo que el cliente recibe los resultados de la ejecución y por tanto no tiene que realizar ninguna acción especial, con lo que el código *PHP* no causa problemas de interpretación o ejecución al cliente. Esto

evita problemas de las diferentes versiones de navegadores que existen, que algunas veces soportan de distinta manera a los lenguajes como JavasScript y VisualScript.



Diag. 4. Interpretación y conversión de código PHP a HTML

En el nivel más básico, *PHP* puede hacer lo que podría hacerse utilizando CGI, obtener datos de una forma, generar páginas con contenido dinámico o recibir y enviar cookies. Tal vez la característica más fuerte y significativa en *PHP* es el soporte de un gran rango de bases de datos, por lo que escribir el código de acceso a las bases no es algo complejo, pero si se deseara cambiar a otro manejador de bases de datos, se tendría que volver a escribir el código que interviene en el manejo de la base de datos. Por ejemplo, para conectarse a una base de datos Sybase se utiliza la instrucción `sybase_connect()`, pero para conectarse a MySql se emplea otra diferente, `mysql_connect()`

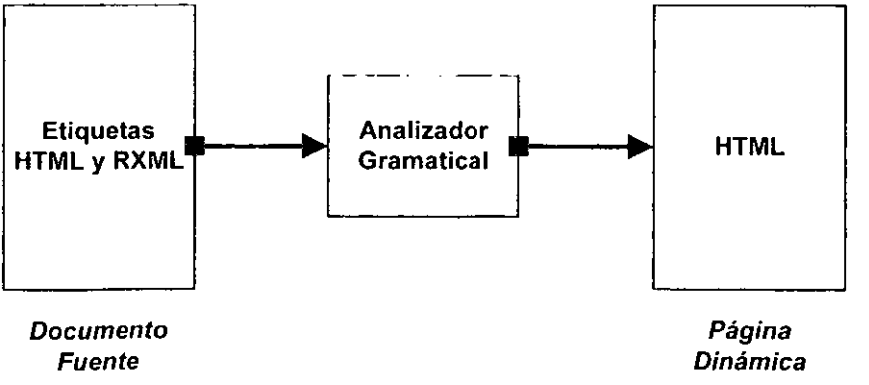
Aquí radica la desventaja de utilizar *PHP*, ya que es imposible cambiar de una manera más transparente entre un manejador de bases de datos y otro.

Los manejadores de bases de datos con los que trabaja *PHP* son: Adabas D, Ingres, Oracle, dBase, InterBase, FrontBase, PostgreSQL, mSQL, Solid, Hyperwave, MS-SQL, Sybase, IBM DB2, MySQL, Velocis, Informix, ODBC.

PHP puede ser empleado con distintos servidores Web, por ejemplo: con Roxen Web Server y Apache Server.

IV.2.2. RXML (Roxen Markup Language; Lenguaje de Marcado de Roxen)

Como se mencionó anteriormente, las páginas Web son escritas usando *HTML*. Roxen WebServer tiene su propio lenguaje, llamado *RXML*, que emplea etiquetas similares a las de *HTML*, sin embargo las etiquetas *RXML* no son enviadas al navegador, dado que Roxen WebServer convierte todas las etiquetas *RXML* en *HTML* usando un analizador. Este esquema es equivalente al de *PHP*, que también utiliza un analizador para generar páginas *HTML*.



Diag. 5. Interpretación y conversión de etiquetas RXML a HTML

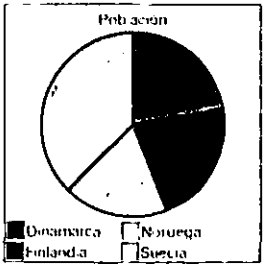
RXML puede ser empleado para diversas tareas, como crear gráficos, diagramas, para conexión a bases de datos a través de distintos mecanismos, o para crear páginas dinámicas.

Cada etiqueta de *RXML* tiene una función específica, que al combinarse con otras etiquetas, permiten resolver tareas más complejas.

RXML está diseñado para ser fácil de aprender, especialmente para la gente con habilidades en el *HTML*, pues su sintaxis y semántica visible son similares al *HTML*. Los elementos de *RXML* siguen las reglas de *XML* para seguir la tendencia hacia la que se está orientando el desarrollo en Web.

A continuación se muestra un ejemplo de código que genera una imagen GIF con una gráfica de pastel a partir de los datos suministrados:

```
<diagram type='pie' width='200' height='200'  name='Población'>
  <data separator=',' >5305048,5137269,4399993,8865051</data>
  <legend separator=',' >Dinamarca,Finlandia,Noruega,Suecia</legend>
</diagram>
```



Diag. 6. Gráfica de pastel generada con RXML

En este ejemplo los datos se introducen como valores fijos, sin embargo estos podrían ser obtenidos a partir de la consulta a una base de datos.

Con *RXML* se pueden crear nuevas etiquetas de acuerdo a las necesidades específicas de cierta tarea. De esta manera el trabajo del diseñador de Web y del desarrollador puede ser separado. El desarrollador puede asociar código a etiquetas *RXML* que entonces se puedan utilizar por los diseñadores de Web.

Por ejemplo, Las siguientes líneas generan una nueva etiqueta llamada saludo que recibe como parámetro el nombre de la persona y opcionalmente el idioma:

```
<define tag="saludo">
  <if variable="&_.lenguaje is Ingles">Hello &_.nombre;!</if>
  <else>Hola &_.nombre;!</else>
</define>
```

Y para utilizarla se ocupa:

```
<saludos nombre="Alberto" />  <!--Imprimirá: Hola Alberto! -->
<saludos nombre="Alberto" lenguaje="Ingles" /> <!--imprimirá: Hello Alberto! -->
```

RXML fue diseñado pensando en seguridad, lo cual permite escribir aplicaciones complejas y seguras. El control de acceso a la aplicación puede controlarse al nivel de directorio, por módulos y por direcciones IP.

A través de *RXML*, es posible conectarse a diversos manejadores de bases de datos. A diferencia de *PHP*, *RXML* tiene etiquetas únicas para realizar la interacción con los distintos manejadores de bases de datos. Esto, brinda la posibilidad de intercambiar de manera más o menos transparente entre los distintos manejadores de bases de datos. El intercambio de manejadores de bases de datos no es totalmente transparente porque aunque siguen la misma lógica del estándar de *SQL*, difieren en algunos características que dependen de la compañía a la que pertenece el *RDMBS*. Si una aplicación realizada con *RXML* se quiere utilizar con un manejador de bases de datos diferente, lo único que habría que revisar, y en su caso modificar, serían las instrucciones *SQL* que se envían al servidor de bases de datos.

Rxml permite interactuar con los siguientes manejadores de bases de datos: Oracle, Sybase, MySQL, MiniSQL, Postgres SQL, Informix, Adabas D, IBM DB2, ODBC.

IV.2.3. Java.

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems en 1991 como parte de un proyecto de investigación para crear software para dispositivos electrónicos (equipos de televisión, videocaseteras, etc.)

Creado con base en el lenguaje de programación C++, *Java* se diseñó para ser pequeño, sencillo y portátil a diversas plataformas. Estas características lo hicieron perfecto para distribuir programas ejecutables vía Web.

Java nos permite desarrollar tanto aplicaciones como *Applets* (los cuales corren del lado del cliente), además actualmente se pueden programar del lado del servidor a través de *Servlets* y *Java* Server Pages.

IV.2.3.1. Applets

Los *Applets* son programas que utilizan una interfaz gráfica que sabe como comunicarse con el navegador de Web. Son programas compilados que son interpretados por la máquina virtual de *Java* que viene incluida con el navegador de Web, por lo tanto, son ejecutados del lado del cliente. *Java* fue muy popular por esta característica, que permitía implementar contenido dinámico en páginas de Web. Sin embargo tiene varios inconvenientes:

- Si la conexión a la página Web es vía módem, la transferencia de los elementos necesarios para comenzar a funcionar será muy lenta, a comparación de código *HTML*.
- Cada navegador de Web implementa alguna versión de la máquina virtual de *Java*, que en ocasiones no es compatible con otras versiones mostrando un funcionamiento inadecuado.
- Actualmente existen mejores tecnologías para generar contenido dinámico del lado del cliente.

En general *Java* se ha orientado hacia desarrollo de aplicaciones para Intranets, ya que en ellas no es importante el tiempo de transferencia de los *Applets* y es más fácil uniformizar a los usuarios para que empleen la misma versión del navegador de Web.

IV.2.3.2. Servlets

Los *Servlets* son archivos *Java* compilados que pueden ser ejecutados por un servidor Web. En funcionalidad, los *Servlets* se encuentran entre los programas del tipo CGI (Common Gateway Interface, Interfaz Común de Entrada) y las extensiones propietarias del servidor tales como el Netscape Server *API* (NSAPI) o los Módulos de Apache.

Los *Servlets* tiene las siguientes ventajas sobre otros mecanismos de extensión del servidor:

- Generalmente son más rápidos que los scripts CGI porque utiliza un modelo de procesos diferente.
- Utilizan un estándar de *API* que es soportado por muchos servidores Web.
- Tiene todas las ventajas del lenguaje de programación *Java*, incluyendo la facilidad de desarrollo e independencia de la plataforma.

- Pueden tener acceso al gran conjunto de *APIs* disponibles para la plataforma *Java*.

Los *Servlets* son empleados en mayor grado en aplicaciones para Internet que los *Applets*, ya que funcionan del lado del servidor y no se tiene que depender en la versión de *Java* instalada en el navegador de Web.

Comprado con *PHP*, *JSP* o RML, los *Servlets* son más complicados de programar y mantener ya que el menor cambio requerido en el formato de la página de Web, involucra la modificación del código del *Servlet* y su recompilación. Un diseñador de páginas de Web, difícilmente tendrá los conocimientos necesarios para modificar el código de *HTML* dentro de un *Servlet*.

IV.2.3.3. JSP

La tecnología de *Java* Server Pages (*JSP*) permite desarrollar y diseñar sitios Web, ricos en información, con páginas Web dinámicas. Funciona del lado del servidor al igual que los *Servlets*.

Como parte de la tecnología *Java*, los *JSPs* permiten un desarrollo rápido de aplicaciones Web que son independientes de la plataforma y además separa la interfaz del usuario de la generación de contenido permitiendo a los diseñadores cambiar la distribución de la página sin alterar el contenido dinámico subyacente, a diferencia de los *Servlets*.

Los *JSPs* utilizan etiquetas como *XML* y scripts escritos en lenguaje de programación *Java* para encapsular la lógica que genera el contenido para la página.

Todas las etiquetas del formato (*HTML* o *XML*) se pasan directamente a la página de respuesta. La tecnología de *Java* Server Pages es una extensión de

la tecnología *Java Servlet*. Juntas, la tecnología *JSP* y los *Servlets* proporcionan una alternativa atractiva a otros tipos de programación dinámica para Web que ofrecen una independencia de la plataforma, mejoramiento del desempeño, separación de la lógica del despliegue y lo más importante, la facilidad de empleo.

Las páginas *JSP* funciona de manera similar a *PHP* y *RXML*, sin embargo, requiere que el programador realice sus propias funciones utilizando *Java*, por ejemplo, es necesario programar funciones para interactuar con la base de datos. Por su parte *PHP* y *RXML* disponen de funciones específicas que realizan este trabajo simplificando la programación.

IV.2.4. JavaScript

Los lenguajes *intérpretes* que vienen incluidos con el navegador de Web, como *JavaScript*, son empleados para realizar algunas verificaciones antes de enviar información al servidor, evitando que se envíen datos erróneos accidentalmente.

El código de *JavaScript* se distribuye con el demás código de *HTML* de la página y es interpretado del lado del cliente por el navegador de Web.

JavaScript es un lenguaje multiplataforma, basado en objetos, y que es interpretado por el navegador de Web, sin embargo existen tecnologías que permiten que *JavaScript* pueda emplearse del lado del servidor.

Los navegadores pueden interpretar el código de *JavaScript* incrustado dentro de un documento *HTML*. Cuando el navegador solicita tal página, el servidor envía el contenido completo del documento, incluyendo *HTML* y las

instrucciones de *JavaScript*, a través de la red al cliente. El navegador muestra el código *HTML* y ejecuta el código de *JavaScript*.

JavaScript es empleado para hacer que la página responda a eventos generados por el usuario, como un clic del ratón.

De las funciones más útiles de *JavaScript* para el desarrollo de aplicaciones para Internet, se tiene la validación de información que el usuario introduce en las formas, sin necesidad de enviar la información al servidor a través de la red. Con *JavaScript*, el cliente puede abarcar parte de la lógica de la aplicación aunque la mayor parte siga residiendo en el servidor.

JavaScript y *Java* son similares en algunos aspectos pero difieren en otros. La sintaxis de las instrucciones de control de flujo son las mismas, sin embargo *Java* requiere de un *compilador* y *JavaScript* requiere de un *Intérprete*. La ventaja de que el código no sea estático, es que ofrece una mayor flexibilidad para trabajar con objetos que son creado en tiempo de ejecución.

Cada navegador implementa su propia versión de *JavaScript*, por lo que pueden llegar a diferir en algunos aspectos. Estas diferencias hacen que dependiendo del tipo de aplicación, sea necesario considerar hasta que grado es factible emplearla. Lo ideal es únicamente emplear las características estándares y no las adicionales que cada compañía implementa.

IV.2.5. VBScript

VBScript es muy similar a *JavaScript*, sin embargo es una tecnología basada en productos Microsoft, dada esta característica, sólo es conveniente emplearla cuando se sepa de antemano que Microsoft Internet Explorer es el único navegador de Web que será empleado para visualizar la página. Esto podría ser factible en una Intranet, donde cada máquina tenga instalada la misma versión del navegador de Web, pero en Internet, dada su extensión, es imposible saber con anticipación que navegador será empleado.

VBScript también puede ser utilizado del lado del servidor con tecnologías Microsoft. Por ejemplo, la tecnología *ASP* permite utilizar tanto *VBScript* como *JavaScript* para generar páginas de contenido dinámico. Estas versiones que funcionan del lado del servidor tienen instrucciones adicionales que permiten establecer una comunicación con las bases de datos y manipular la información que de ella se recupere. El problema con la tecnología *ASP* es que está diseñada para funcionar en plataforma Windows únicamente y con el servidor Microsoft Internet Information Server. Estas restricciones nos limitan a cierto tipo de equipo y de software que se puede emplear para desarrollar páginas dinámicas, por lo que no se adentrará en detalles de la tecnología *ASP*.

VBScript también es un lenguaje que es interpretado, y viene disponible en el navegador Microsoft Internet Explorer.

V. CASO DE ESTUDIO: CREACIÓN DEL BANCO DE INFORMACIÓN VETERINARIA EN INTERNET

V.1. ¿Qué es el Banco de información Veterinaria?.

EL Banco de Información Veterinaria (BIVE) se creó en 1985 con el propósito de localizar, analizar y difundir la información que se produce en América Latina y el Caribe sobre medicina veterinaria y zootecnia.

BIVE cubre todas las áreas relacionadas con la medicina veterinaria y zootecnia: acuacultura, administración, anatomía, apicultura, aseguramiento de la calidad de los productos y subproductos pecuarios, bioestadística, bioquímica, caprinocultura, cirugía, citología, clínica, cunicultura, ecología, embriología, enfermedades infecciosas, entomología, epidemiología, epizootiología, extensionismo, farmacología, fauna silvestre, fisicoquímica, fisiología, genética, higiene, inmunología, legislación veterinaria, manejo de forrajes, mercadotecnia, micología, microbiología, nutrición, ovinocultura, parasitología, patología, pequeñas especies, producción animal, reproducción, salud publica, sociología, terapéutica, toxicología, virología, zoonosis.

BIVE reúne la información proveniente de publicaciones periódicas, tesis, monografías y literatura no convencional publicadas a partir del año de 1984 contando con aproximadamente 14,000 registros.

V.2. Problemática.

La información del BIVE, originalmente se capturó empleando un software desarrollado en la UNESCO, llamado Micro CDS/ISIS. Este programa Inicialmente estaba disponible para *DOS*, y últimas versiones ya se encuentran

disponibles para Windows. Cuando surgió la primera versión, en diciembre de 1985 era la mejor solución para manejar información de referencias, ya que las aplicaciones que manipulaban registros de tamaño variable, como DBASE III, desperdiciaban una cantidad considerable de espacio al no poder manejarlos de la manera óptima y dado el alto costo de almacenamiento en esos años, no era algo que se pudiera desperdiciar.

Únicamente los manejadores de bases de datos relacionales eran capaces de manipular de manera óptima la información, pero eran muy caros y los sistemas de hardware en los que funcionaban también lo eran por lo que sólo estaban a disposición de grandes corporaciones.

Micro CDS/ISIS, permitía manejar la información referencial, de una manera adecuada, empleando computadoras personales XT, que eran las más accesibles por su precio. Sin embargo el esquema "no relacional" que utiliza para almacenar y recuperar información es obsoleto y poco operacional a pesar de las mejoras que se han implementado en dicho software. Han pasado 15 años en los que la tecnología ha crecido constantemente y los avances en manejadores de bases de datos han sido impresionantes, opacando las ventajas que en su momento hizo ser a Micro CDS/ISIS la mejor opción para manejo de este tipo de información.

¿Cuál es el problema de manejar un esquema "no relacional" para el manejo de información? El problema radica en que se tiene redundancia de información, es decir, se encuentra duplicada generando desperdicio de espacio de almacenamiento, disminución en el rendimiento de las consultas e inconsistencias de información.

Pero el problema principal de emplear un modelo "no relacional" se encuentra en que la tecnología actual para desarrollar aplicaciones, se basa

principalmente en la metodología relacional con su lenguaje de consulta estándar *SQL* (Strutured Query Language, Lenguaje Estructurado de Consulta). Por tanto el uso de una metodología que no es relacional, implica severas limitaciones en cuanto al número de tecnologías disponibles y cantidad de personas que pueden dar mantenimiento ó iniciar nuevos proyectos.

La otra ventaja de Micro CDS/ISIS, era con respecto al precio. Micro CDS/ISIS puede ser obtenido sin costo alguno, sin embargo no se distribuye el código fuente del software, únicamente el binario compilado, mientras que en la actualidad existen manejadores de bases de datos muy eficientes que se encuentran bajo licencia *GPL*, es decir, que además de poderse obtener sin cargo alguno, el código fuente está a disposición del usuario, el cual puede adaptarlo a sus necesidades particulares y funciona bajo diversas plataformas. Además existen versiones de bases de datos comerciales que son gratuitas con fines de desarrollo.

Por las razones expuestas anteriormente se puede afirmar que la tecnología empleada con Micro CD/ISIS es obsoleta por lo que es necesario emplear una nueva que permita adaptarse mejor a la tecnología de Internet y estándares para consulta y manejo de información.

V.3. Desarrollo.

V.3.1. Propósito del sistema.

El sistema debe manejar y consultar la información del Banco de Información Veterinaria, poniendo a disposición de un mayor número de personas, dicha información a través de Internet. Además debe facilitar el mantenimiento y consulta de información manteniendo a esta en un esquema centralizado. Contar con una interfaz gráfica sencilla es de vital importancia, para que cualquier persona pueda sacar el máximo provecho del BIVE.

V.3.2. Proceso actual para el manejo del BIVE.

Actualmente la información se almacena en una o varias máquinas mediante el programa CD/Isis. Éste, debe estar instalado en cada computadora que se desee utilizar para realizar consultas y para dar mantenimiento a la información, siendo independiente ésta entre computadoras.

Como se desea publicar esta información en el Web, es necesario sacarla de cada una de las computadoras donde se haya realizado la captura, posteriormente se requiere convertirla a un formato especial, llevarla al servidor y correr algunos procesos para dar funcionalidad de búsqueda. Finalmente se podría emplear el programa wwwisis para realizar consultas a la información. Sin embargo wwwisis no utiliza un lenguaje estándar de consulta lo que complica su uso.

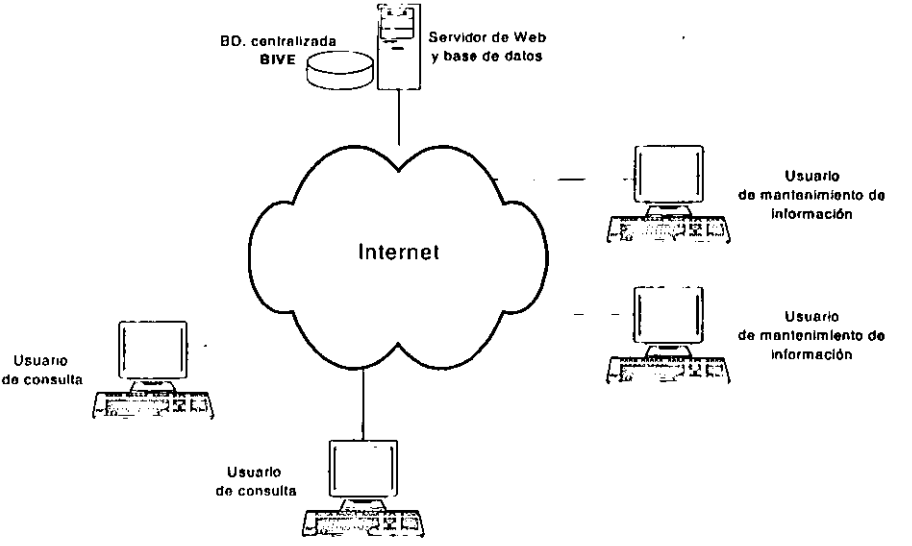
Pero el problema principal radica en que toda esta serie de pasos dificulta tener un control sobre la información, alentando severamente el proceso de mantenimiento y publicación de dicha información en el Web.

V.3.3. Propuesta.

Para que el sistema cumpla con el propósito especificado, se realizará lo siguiente:

- Centralización de la información utilizando una base de datos relacional.
- *Normalización* de la información existente para poder integrarla al esquema relacional.

- El acceso a la información será posible, prácticamente desde cualquier lugar, a través de un navegador de Web.
- La publicación de la información conforme se vaya capturando será inmediata, sin necesidad de realizar procesos adicionales siguiendo el siguiente esquema:

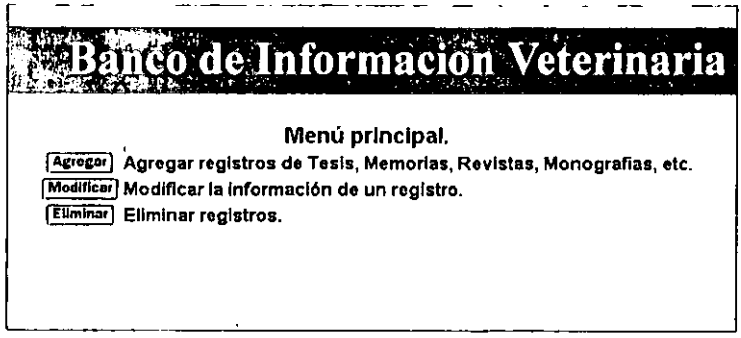


Diag. 7. Propuesta del sistema

El sistema se dividirá en dos módulos: ***Administración*** y ***Consulta*** de la información.

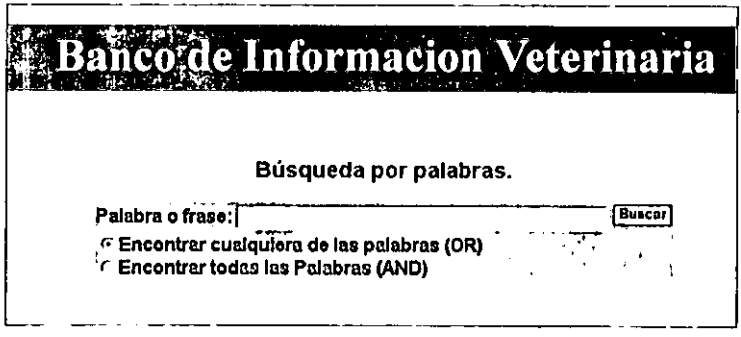
El módulo de administración permitirá realizar las siguientes operaciones sobre la información:

- Agregar nuevos registros al BIVE.
- Modificar la información en caso de ser necesario.
- Eliminar la información no deseada.

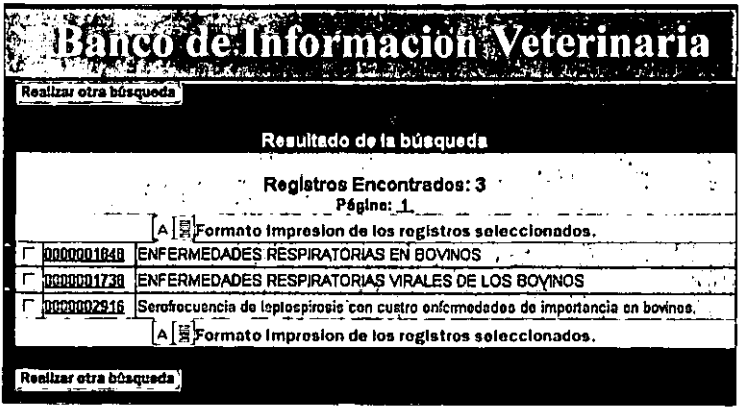


Diag. 8. Menú del módulo de administración.

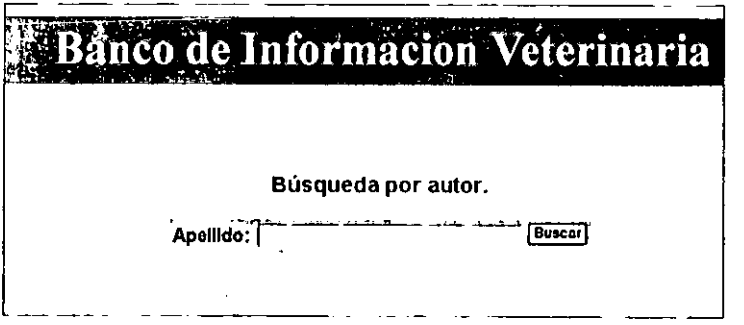
El módulo de consulta brindará dos métodos para realizar las búsquedas en el BIVE: palabras o autor. En la búsqueda por palabras será posible especificar si se desea encontrar información que incluya todas las palabras especificadas o cualquiera de ellas (operadores AND y OR respectivamente).



Diag. 9. Módulo de consulta - búsqueda por palabras.



Diag. 10. Módulo de consulta – resultado de la búsqueda por palabras.



Diag. 11. Módulo de consulta - búsqueda por autor.

Diag. 12. Módulo de consulta – resultado de la búsqueda por autor.

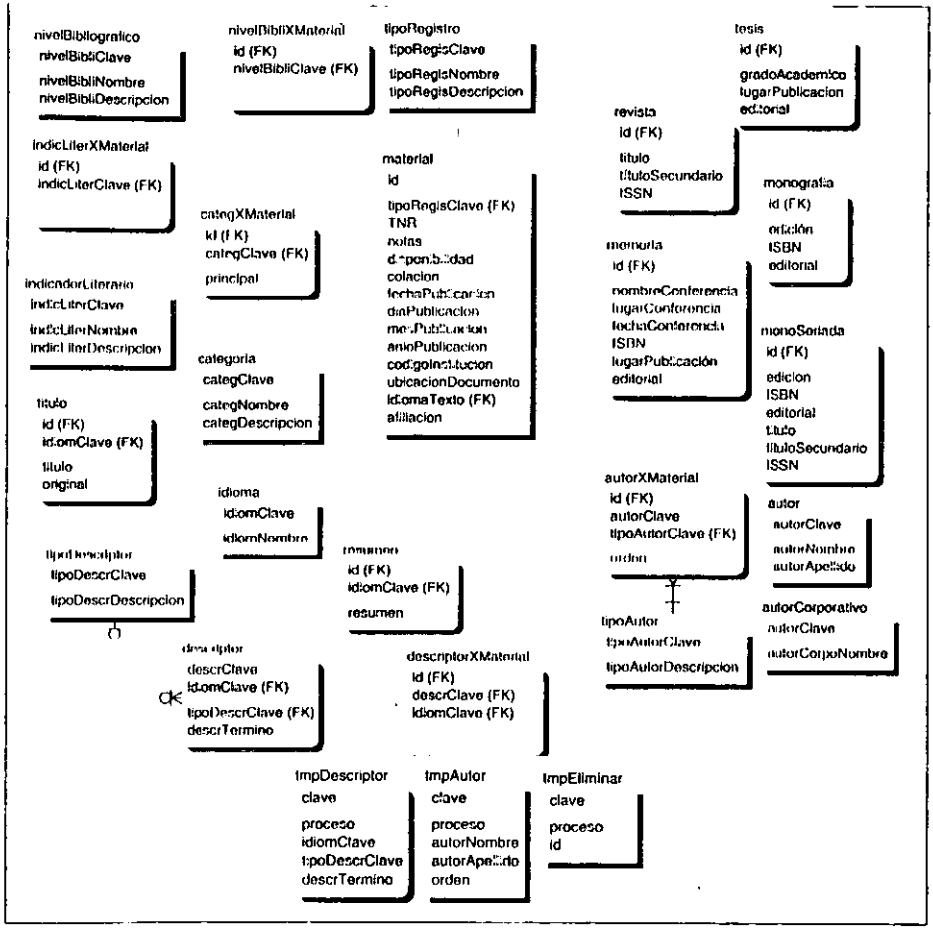
Tener la información centralizada permite que varios usuarios capturen la información al mismo tiempo, sin necesidad de procesos posteriores para concentrar la información. La aplicación será capaz de manejar la concurrencia de usuarios tanto en el módulo de consulta como en el de administración.

Para desarrollar la aplicación se elegirán las herramientas más adecuadas tomando en consideración los recursos de hardware y software disponibles en la dependencia, prefiriendo a aquellas que sean flexibles y escalables para evitar que la aplicación se vuelva obsoleta rápidamente.

El primer módulo a desarrollar será el de consulta por ser el más importante, y se proseguirá con el de administración para que se puedan agregar nuevos registros.

V.3.4. Diseño de la base de datos.

V.3.4.1. Diagrama Entidad/Relación.



V.3.4.2. Diccionario de datos.

(PK)= llave primaria (FK) = llave foránea

Nombre:	**descriptor**		
Descripción:	Esta tabla contiene los descriptores que se aplican a los materiales		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
descrClave **(PK1)**	Clave del descriptor	int(10)	No
idiomClave **(FK, PK1)**	Clave del idioma para el descriptor	char(2)	No
tipoDescrClave **(FK)**	Clave del tipo de descriptor	char(1)	Si
descrTermino	Descriptor de tema.	varchar(100)	Si

Ejemplos:

descrClave	**idiomClave**	**tipoDescClave**	**descrTermino**
1	ES	T	ABEJA
2	ES	T	ABONOS FOSFATADOS

Nombre:	**autor**		
Descripción:	Esta tabla contiene a los autores personales		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
autorClave **(PK)**	Clave del autor	int(10)	No
autorNombre	Nombre del Autor Personal	varchar(50)	Si
autorApellido	Apellidos Paterno y Materno del Autor	varchar(100)	No

Ejemplos:

autorClave	**autorNombre**	**autorApellido**
1	C. I.	Alvarez M.
2		Alvarez Martínez
3	P.	William

Nombre:	autorXMaterial		
Descripción:	Esta tabla relaciona a los autores personales o corporativos con el material		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK1)	Clave única para identificar al material	int(10)	No
autorClave (FK2, PK1)	Clave del autor Personal o corporativo	int(10)	No
tipoAutorClave (FK2, PK3)	Clave del tipo de autor Unicamente existen 2 tipos P Personal C Corporativo	char(1)	No
orden	Orden de importancia de los autores involucrados	tinyint(4)	Si

Ejemplos:

id	autorClave	orden
1	1	1
1	2	2
3	2	

Nombre:	categoria		
Descripción:	Esta tabla contiene las categorías con las que se clasifican los materiales		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
categClave (PK)	Clave de la categoría. Estas son propuestas por la FAO.	char(3)	No
categNombre	Nombre de la categoría	varchar(50)	No
categDescripcion	Descripción de la categoría	varchar(255)	Si

Ejemplos:

categClave	categNombre	categDescripcion
T10	Enfermedades profesionales y riesgos laborales	
S01	Nutrición humana	Aspectos generales de la nutrición humana

Nombre:	categXMaterial		
Descripción:	Esta tabla relaciona las categorías con los materiales		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK1)	Clave única para identificar al material	int(10)	No
categClave (FK, PK1)	Clave de la categoría. Estas son propuestas por la FAO.	char(3)	No
principal	Indica si es una categoria principal o no	char(1)	Si

Ejemplos:

Id	categClave	principal
1	S01	S
1	T10	
2	T10	S

Nombre:	autorCorporativo		
Descripción:	Esta tabla contiene a los autores corporativos		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
autorClave (PK)	Clave del autor corporativo	int(10)	No
autorCorpoNombre	nombre del autor corporativo	text (400)	No

Ejemplos:

autorClave	autorCorpoNombre
1	UNIVERSIDADE ESTADUAL DE LONDRINA

Nombre:	idioma		
Descripción:	Esta tabla contiene los idiomas		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
idiomClave (PK)	Clave del idioma	char(2)	No
idiomNombre	Idioma.	varchar(40)	No

Ejemplos:

idiomClave	idiomNombre
Es	Español
En	Inglés
Pt	Portugués

Nombre:	**indicadorLiterario**		
Descripción:	Esta tabla contiene los indicadores literarios que puede tener un material		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
indicLiterClave **(PK)**	Clave del indicador de literatura	char(1)	No
indicLiterNombre	Nombre del indicador Literario	varchar(30)	No
indicLiterDescripcion	Descripción del indicador Literario	varchar(255)	Si

Ejemplos:

indicLiterClave	**indicLiterNombre**	**indicLiterDescripcion**
K	conferencia	
U	tesis o disertación	En esta categoria entran todos aquellos materiales que estén relacionados con una tesis
V	No convencional	

Nombre:	**indicLiterXMaterial**		
Descripción:	Esta tabla relaciona los indicadores literarios con los materiales		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id **(FK, PK1)**	Clave única para identificar al material	int(10)	No
indicLiterClave **(FK, PK1)**	Clave del indicador de literatura.	char(1)	No

Ejemplos:

id	**indicLiterClave**
1	K
1	U
2	V

Nombre:	monoSeriada		
Descripción:	Esta tabla contiene los datos particulares de las monografías seriadas		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK)	Clave única para identificar al material	int(10)	No
edicion	Edición de la monografía seriada	varchar(80)	Si
ISBN	ISBN de la monografía seriada	varchar(80)	Si
editorial	Editorial de la monografía seriada	varchar(255)	Si
titulo	titulo principal de la monografía seriada	text (400)	Si
tituloSecundario	titulo secundario de la monografía seriada	text (400)	Si
ISSN	ISSN de la monografía seriada	varchar(80)	Si

Ejemplos:

id	edicion	ISBN	editorial	titulo	titulo Secundario	ISSN
4				Salud Animal		1436-4522
6	2ª	90-70002-34-5	MacGraw			

Nombre:	material		
Descripción:	En esta tabla se encuentran los datos comunes a todos los tipos de materiales, como son tesis, memorias, revistas, monografías y libros		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (PK)	Clave única para identificar al material	int(10)	No
tipoRegisClave (FK)	Clave del tipo de registro	char(1)	No
TNR	Numero Temporal de Registro. Identificador que se le asigna al registro para control interno	char(9)	Si
idiomaTexto (FK)	Clave del idioma del texto	char(2)	No
paisPublicacion	Clave del país donde se publicó el material.	char(2)	Si
anioPublicacion	Año de publicación del material	varchar(4)	Si
mesPublicacion	Mes de publicación del material	char(2)	Si
diaPublicacion	Día de publicación del material	char(2)	Si
codigoInstitucion	Código de la institución.	varchar(10)	Si
Afiliacion	Datos para contactar a los autores.	varchar(255)	Si
colación	Páginas donde se encuentra la referencia	varchar(40)	Si
notas	Notas acerca del material	varchar(160)	Si
disponibilidad	Donde se encuentra el material, dirección completa	varchar(160)	Si
ubicacionDocumento	Departamento donde se encuentra el documento	varchar(40)	Si

Ejemplos:

Id	tipo Regis Clave	TNR	Idioma Texto	anio Publicacion	mes Publicacion	dia Publicacion	codigo Institucion
4	B	MX69	Es	2000	05	24	BFMVZ
6	H	MX51	Pt	1999	04	12	

afiliacion	colacion	notas	disponibilidad	ubicación Documento
UNAM. Clínica de pequeñas especies	14-16	Numerical data, 4 ref	UNAM. Circuito escolar S/N. FMVZ	Depto. de prevención.
UNAM. Clínica de pequeñas especies	30-56	Numerical data		

Nombre:	**memoria**		
Descripción:	Esta tabla contiene los datos particulares del material de memorias		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id **(FK, PK)**	Clave única para identificar al material	int(10)	No
nombreConferencia	Nombre de la conferencia	varchar(200)	Si
lugarConferencia	Lugar de la Conferencia	varchar(215)	Si
fechaConferencia	Fecha de la conferencia	varchar(80)	Si
ISBN	ISBN	varchar(80)	Si
lugarPublicacion	Lugar de publicación	varchar(210)	Si
editorial	Editorial	varchar(255)	Si

Ejemplos:

id	nombre Conferencia	lugar Conferencia	fecha Conferencia	ISBN	Lugar Publicacion	Editorial
69	5a. REUNION ANUAL DE PARASITOLOGIA VETERINARIA	México	del 5 al 10 de mayo del 2000	92-70202-34-8	México	

Nombre:	monografia		
Descripción:	Esta tabla contiene los datos particulares del material de monografías		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK)	Clave única para identificar al material	int(10)	No
edicion	Edición de la monografía	varchar(80)	Si
ISBN	ISBN de la monografía	varchar(80)	Si
editorial	Editorial de la monografía	varchar(255)	Si

Ejemplos:

id	edicion	ISBN	editorial
70	Primera Edición	90-70002-34-5	McGrawHill
75	1era Ed.	90 70121 25 5	McGrawHill

Nombre:	nivelBibliografico		
Descripción:	Esta tabla contiene los niveles bibliográficos que puede tener un material		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
nivelBibliClave (PK)	Clave del Nivel Bibliográfico	char(1)	No
nivelBibliNombre	Nombre del nivel bibliográfico	varchar(20)	No
nivelBibliDescripcion	Descripción del nivel bibliográfico	varchar(255)	Si

Ejemplos:

nivelBibliClave	nivelBibliNombre	nivelBibliDescripcion
S	Seriado	Se aplica a todas las publicaciones que se emitan periódicamente
M	Monográfico	
A	Analítico	

Nombre:	nivelBibliXMaterial		
Descripción:	Esta tabla relaciona los niveles bibliográficos con el material		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK1)	Clave única para identificar al material	int(10)	No
nivelBibliClave (FK, PK1)	Clave del Nivel Bibliográfico	char(1)	No

Ejemplos:

id	nivelBibliClave
1	A
1	M
1	S
2	A
2	S
3	S

Nombre:	resumen		
Descripción:	Esta tabla contiene los resúmenes que pueda tener el material		
Columna	Descripción	Tipo de dato	Nulo
id (FK,PK1)	Clave única para identificar al material	int(10)	No
idiomClave (FK,PK1)	Clave del idioma	char(2)	No
resumen	Resumen del material	text	No

Ejemplos:

id	idiomClave	resumen
1	Es	Se realizó un análisis de la información pecuaria mexicana, desde 1876 a 1996. Se revisaron las fuentes primarias que fue posible localizar y se recurrió a fuentes secundarias para ...

Nombre:	revista		
Descripción:	Esta tabla contiene los datos particulares del material de las revistas		
Columna	Descripción	Tipo de dato	Nulo
id (FK, PK)	Clave única para identificar al material	int(10)	No
titulo	Título principal de la revista	text (400)	Si
tituloSecundario	Título secundario de la revista	text (400)	Si
ISSN	ISSN de la revista	varchar(80)	Si

Ejemplos:

id	titulo	tituloSecundario	ISSN
1	Revista Veterinaria	Pequeñas especies	1436-4525

Nombre:	descriptorXMaterial		
Descripción:	Esta tabla relaciona los descriptores con los materiales		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK1)	Clave única para identificar al material	int(10)	No
descrClave (FK2, PK1)	Clave del descriptor.	int(10)	No
idiomClave (FK2, PK1)	Clave del idioma de descriptor	char(2)	No

Ejemplos:

id	descrClave	idiomClave
1	1	Es
1	2	Es
1	4	Es
2	1	Es

Nombre:	tesis		
Descripción:	Esta tabla contiene los datos particulares del material de tesis		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK)	Clave única para identificar al material	int(10)	No
gradoAcademico	Grado Académico del que presenta la tesis	varchar(80)	Si
lugarPublicacion	Lugar de la publicación.	varchar(210)	Si
editorial	Editorial de la tesis (Facultad del tesista)	varchar(255)	Si

Ejemplo

id	gradoAcademico	lugarPublicacion	editorial
78	Licenciatura	México DF	Universidad Nacional Autónoma de México. FMVZ
79	Maestria	Colima	Universidad de Colima. FMVZ

Nombre:	tipoAutor		
Descripción:	Esta tabla contiene los tipos de autor posibles		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
tipoAutorClave (PK)	Clave del tipo de autor Unicamente existen 2 tipos P Personal C Corporativo	char(1)	No
tipoAutorDescripcion	Descripción del tipo de Autor	varchar(20)	No

Ejemplo

tipoAutorClave	tipoAutorDescripcion
P	Personal
C	Corporativo

Nombre:	tipoDescriptor		
Descripción:	Esta tabla contiene los tipos de descriptores posibles		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
tipoDescrClave (PK)	Clave del tipo de descriptor	char(1)	No
tipoDescrDescripcion	Tipo de descriptor. Sólo existen 3: L Locales, T Tesauro y P Propuestos	varchar(100)	No

Ejemplos:

tipoDescrClave	tipoDescrDescripcion
L	Local
T	Tesauro
P	Propuesto

Nombre:	tipoRegistro		
Descripción:	Esta tabla contiene los tipos de materiales posibles		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
tipoRegisClave (PK)	Clave del tipo de registro	char(1)	No
tipoRegisNombre	Nombre del tipo de registro	varchar(50)	No
tipoRegisDescripcion	Descripción del tipo de registro	varchar(255)	Si

Ejemplos:

tipoRegIsClave	tipoRegIsNombre	tipoRegIsDescripcion
B	Monografía	
P	Patente	Abarca cualquier tipo de patente en el ámbito de la medicina veterinaria
D	Dibujo	

Nombre:	titulo		
Descripción:	Esta tabla contiene los títulos de los materiales.		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
id (FK, PK1)	Clave única para identificar al material	int(10)	No
idiomClave (FK, PK1)	Clave del idioma	char(2)	No
titulo	Título del libro.	text	No
original	Indica si el titulo esta en el idioma original (S)	char(1)	Si

Ejemplos:

id	idiomClave	titulo
1	Es	EFECTO DE BORDES EN ENSAYOS DE RENDIMIENTO EN SORGO
1	En	EPIDEMIOLOGY OF RUMINANT GASTROENTERITIS
2	En	TRANSFER FACTOR IMMUNOTHERAPY IN CLINICALLY SICK LACTATING CALVES

Nombre:	tmpAutor		
Descripción:	La información de los autores personales contenida en esta tabla es temporal y se emplea en el proceso de captura únicamente.		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
clave (PK)	Clave del registro temporal (con autoincremento).	bigint(20)	No
proceso	Clave de proceso, empleada para evitar problemas de concurrencia en el proceso de captura	bigint(20)	No
autorNombre	Nombre del autor personal	varchar(50)	No
autorApellido	Apellidos del autor personal	varchar(100)	No
orden	Campo para preservar el orden de captura de los autores.	tinyint(4)	No

Nombre:	tmpDescriptor		
Descripción:	La información de los descriptores contenida en esta tabla es temporal y se emplea en el proceso de captura únicamente.		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
clave (PK)	Clave del registro temporal (con autoincremento).	bigint(20)	No
proceso	Clave de proceso, empleada para evitar problemas de concurrencia en el proceso de captura	bigint(20)	No
idiomClave	Clave del idioma del descriptor	char(2)	No
descrTermino	Descriptor de tema.	varchar(100)	No

Nombre:	tmpEliminar		
Descripción:	Las claves de material contenidas en esta tabla son temporales y se emplean unicamente en el proceso de eliminación de registros.		
Columna	**Descripción**	**Tipo de dato**	**Nulo**
clave (PK)	Clave del registro temporal (con autoincremento).	bigint(20)	No
proceso	Clave de proceso, empleada para evitar problemas de concurrencia en el proceso de captura	bigint(20)	No
id	Identificador del material	int(10)	No

V.4. Elección de plataforma de trabajo y herramientas.

V.4.1. Sistema Operativo.

El sistema elegido fue *Linux* que es de tipo *Unix*, dado que es libre lo cual ahorra gastos en el desarrollo, además ha obtenido una gran aceptación en el mundo de Internet, lo que asegura que existe un soporte amplio brindado tanto por compañías especializadas como por la extensa comunidad de usuarios *Linux* alrededor de todo el mundo. Otra razón para elegir *Linux*, radicó en que el equipo de cómputo otorgado por la facultad de Medicina Veterinaria y Zootecnia para el desarrollo de la aplicación era una computadora personal (Pentium III a 500MHz, 64 MB en RAM).

Linux tiene todas las ventajas de un *Unix* comercial, y si en algún momento se necesita o se desea migrar la aplicación a los equipos de Sun Microsystems con sistema operativo Solaris, equipo con el que cuenta ya la facultad, el cambio será transparente, aun cuando la arquitectura entre una computadora personal y un servidor de Sun Microsystems difiera completamente, y el sistema operativo sea en el último caso Solaris, el cual es de tipo *Unix*.

V.4.2. Servidor de páginas Web.

Dentro de las dependencias de la UNAM, el servidor de páginas de Web que predomina es Apache Web Server, sin embargo, en este trabajo se decidió trabajar con Roxen Web Server, cuyas características invitan a evaluarlo como una alternativa al Apache Web Server, tratando de mostrar sus ventajas y desventajas.

Roxen Web Server permite emplear distintas tecnologías para desarrollo, y eso lo hace muy flexible, pudiendo desarrollar e integrar proyectos con distintas tecnologías utilizando el mismo servidor de Web.

V.4.3. Herramientas de programación.

Roxen permite trabajar con *JSP*, *Servlets*, *PHP*, *RXML* y Perl, lo cual permite tener una gran colección de herramientas de donde elegir.

La herramienta de programación elegida para su empleo del lado del servidor fue *RXML*, dada la flexibilidad que brinda para desarrollar prototipos. Una de las razones principales por la que *RXML* fue elegida como herramienta de programación, radicó en su facilidad para comunicarse con distintas bases de datos, sin necesidad de cambiar instrucciones de *RXML*.

PHP hubiera sido también una buena elección, sin embargo, al comenzar el proyecto todavía no se especificaba cual sería el RDMBS con el que se trabajaría, provocando que si en un momento dado el *RDMBS* tuviera que cambiarse, hubiera sido necesario la reprogramación de la aplicación utilizando las instrucciones específicas para dicho *RDMBS*.

El módulo de administración de Información de BIVE, al ser únicamente utilizado por un pequeño grupo de personas, pudo utilizar cualquier lenguaje presente en el navegador, como *VBScript* o *JavaScript* para llevar la lógica del lado del cliente. Dado que es un grupo pequeño los usuarios de este módulo, es factible condicionar el uso de un navegador específico para poder utilizar la aplicación. En cambio, para el módulo de consulta, no es posible, condicionar al usuario a utilizar un navegador de Web específico, porque se desea que esté a disposición del mayor número de personas, no sólo de aquellos que tengan la última versión de un determinado navegador de Web.

La herramienta elegida fue JavasScript, dado que tanto Internet Explorer como Netscape Navigator lo soportan, aunque con algunas diferencias. JavasScript fue empleado básicamente para realizar validaciones en las formas de captura para prevenir equivocaciones u omisiones de los usuarios.

V.4.4. Manejador de bases de datos.

El manejador de bases de datos es de los elementos principales de la aplicación, ya que su objetivo principal es recuperar la información de manera eficiente, es por eso que fue de las elecciones que más tiempo tomaron, para poder analizar las características de las cuales se podría sacar provecho para el desarrollo de la aplicación. Las bases que se contemplaron fueron Sybase, Oracle y MySQL. Las características que brinda Oracle son muy superiores a las de Sybase y MySQL, sin embargo requiere de gran cantidad de recursos y aunque existe una versión gratuita para fines de desarrollo, no es de distribución libre y es necesario pagar una licencia para su uso en un sistema de producción.

Se consideró que el departamento de informática encargado del BIVE no estaría en la posibilidad de adquirir una licencia de alto costo como la de Oracle ó Sybase para comenzar el desarrollo de la aplicación, por lo cual, se eligió MySQL que es de distribución libre y se obtiene sin costo alguno.

MySQL en su versión actual (3.x) no permite realizar subconsultas (Consultas dentro de otras consultas en *SQL*) lo cual complica en cierto grado el desarrollo de la aplicación, pero brinda otras similares a las de Oracle que son ideales para el manejo de información en Internet. Otra limitante importante de MySQL radica en que no existen objetos en la base de datos que mantengan la integridad referencial, por lo cual es necesario llevar a cabo una programación muy cuidadosa para que la lógica de la aplicación mantenga dicha integridad. A diferencia de MySQL, la mayoría de los *RDMBS* comerciales proveen "constraints" (restricciones), "rules" (reglas) y "triggers" (disparador de procesos) para manejar la integridad de la información.

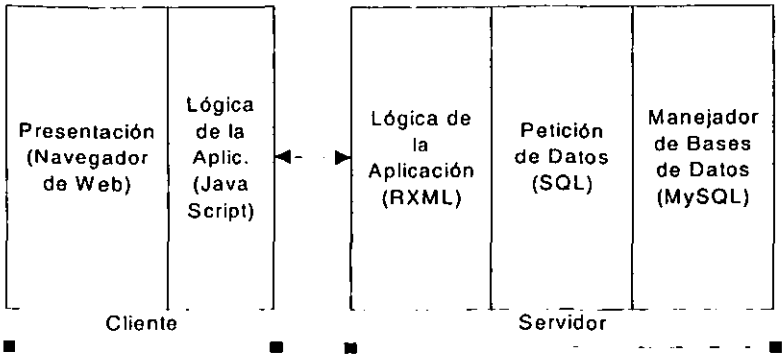
El lenguaje de definición de datos (DDL) de MySQL es muy flexible ya que permite agregar, borrar o cambiar la definición de un campo dentro de una tabla existente, a diferencia de Adaptive Server de Sybase o SQL Server de Microsoft, en los cuales es fundamental contar adicionalmente con herramientas comerciales costosas que sean capaces de realizar estos cambios de manera sencilla.

V.4.5. Arquitectura Cliente/Servidor.

También fue necesario determinar el modelo de arquitectura cliente/servidor a emplear.

- La capa de presentación está totalmente cubierta por el navegador de Web por lo que radica completamente del lado del cliente.
- La lógica de la aplicación está distribuida tanto en el servidor como en el cliente. Del lado del cliente, la lógica de aplicación es mínima ya que únicamente se realizan validaciones utilizando *JavaScript*. En el lado del servidor se encuentra la mayor parte de la lógica de la aplicación, y es donde se emplea *RXML* para poder procesar la información recibida por parte del cliente.
- La petición de datos se lleva a cabo en el mismo servidor donde radica la lógica de la aplicación. *RXML* es empleado para construir y enviar las consultas de *SQL* al servidor de bases de datos.
- El servidor de bases de datos también se encuentra en el mismo servidor donde radica la lógica y petición de datos.

La arquitectura Cliente/Servidor utilizada fue de dos capas con servidor de transacciones, aunque con una pequeña variante, porque parte de la lógica radica, aunque sea mínima, radica en el cliente.



Diag. 13. Variante de la arquitectura Cliente/Servidor de dos capas con lógica repartida entre servidor y cliente.

V.5. Problemas presentados durante el desarrollo de la aplicación.

Dentro de los problemas presentados a lo largo del desarrollo de la aplicación se encontraron los siguientes:

El transformar la información existente al modelo relacional involucró un gran trabajo, dada la redundancia e inconsistencia de la información, por lo que se tuvieron que revisar uno por uno los registros correspondientes a autores para *normalizar* dicha información (La *normalización* es un proceso a través del cual se organiza la información de tal manera que se minimice la redundancia).

Otro problema fue el formato con el cual se identifican a los autores en algunos materiales, ya que es insuficiente e inconsistente para poder evitar la redundancia de información.

En muchas revistas, por ejemplo, se coloca únicamente el primer apellido del autor y un nombre o únicamente las iniciales. Este es un problema que no se puede eliminar salvo en contadas excepciones donde el nombre del autor es bien conocido en la comunidad veterinaria de la Facultad de Medicina Veterinaria y Zootecnia (FMVZ) y por lo tanto se puede completar la información faltante.

VI. CONCLUSIÓN Y DISCUSIÓN DE RESULTADOS

La aplicación del "Banco de Información Veterinaria (BIVE) en Internet" se desarrolló de manera exitosa, y fue posible mantener un costo bajo, eligiendo las herramientas adecuadas, sin comprometer la calidad ni flexibilidad de la aplicación final. El departamento de informática encargado de mantener el BIVE, quedó bastante satisfecho con el sistema y planea integrar información histórica veterinaria (VetHi) bajo el mismo esquema.

Para poder desarrollar aplicaciones que funcionen en Internet no es requisito contar con software ni equipo de cómputo costosos. El desarrollo de la aplicación del BIVE se realizó utilizando como servidor una computadora personal Pentium III a 500Mhz, 64 MB en RAM y con sistema operativo *Linux*, el cual puede conseguirse mediante Internet sin cargo alguno o se puede comprar un disco compacto a un costo muy bajo a comparación de los sistemas operativos de Microsoft.

La disponibilidad del Banco de Información Veterinaria en Internet, hace posible que cualquier usuario pueda tener acceso a la información desde cualquier navegador de páginas de Internet.

La realización de la aplicación fue posible por presentarse como un desarrollo que no requeriría de muchos recursos, de lo contrario, difícilmente hubiera sido apoyada. La aplicación finalizada del "BIVE en Internet", puede brindar elementos para justificar la compra de un equipo de mejor desempeño que soporte mayor número de usuarios y fomentar el desarrollo de aplicaciones similares. Es por eso que el sistema operativo y las herramientas se eligieron pensado en posibles migraciones hacia equipos de cómputo más potentes.

El entorno universitario en el que se desarrolló la aplicación facilitó mantener un costo muy bajo de desarrollo, porque se realizó una propuesta completa en la que el usuario no impuso restricciones para la elección de sistema operativo ni de las herramientas a emplear, sin embargo existen factores que pueden hacer que el costo no se pueda reducir notablemente.

En ocasiones el cliente no confía en el software de distribución libre, por desconocer su calidad y filosofía, prefiriendo el software comercial. En estos casos, hay que proponer la utilización de herramientas comerciales que estén disponibles gratuitamente o con un costo mínimo al ser empleadas con fines de desarrollo. Existen prestigiadas compañías como Oracle, Sybase, Sun Microsystems, etc., que trabajan con este esquema, así que es sólo cuestión de buscar las herramientas adecuadas.

Es importante recordar que la ingeniería de software contempla una etapa previa al de desarrollo: el de definición del sistema. No se debe olvidar que de realizarse un análisis pobre, el costo y tiempo de desarrollo de la aplicación se elevará considerablemente, por constantes modificaciones y adaptaciones que hubieran sido evitadas con un buen trabajo en la etapa de definición.

VII. APÉNDICE.

A. Glosario.

- ***API.*** (Application Program Interface; Programa de aplicación de interfaz). Conjunto de rutinas, protocolos, y herramientas para crear software de aplicación. Un buen *API* facilita el desarrollo de programas porque ofrece al programador una vista simplificada de la aplicación. Aunque las *APIs* están diseñadas para los programadores, éstas son buenas para los usuarios porque garantizan que todos los programas que usan un *API* común, tendrán una interfaz similar, haciendo más sencillo el aprendizaje de nuevos programas a los usuarios.

- ***Analizador gramatical.*** Programa que analiza el código fuente de un programa y lo separa para que pueda procesarse y generar otro tipo de código.

- ***ANSI.*** (American Nacional Standars Institute; Instituto nacional americano de estándares) Sociedad de ingeniería que establece estándares.

- ***Applet.*** pequeño programa incrustado dentro de las páginas *HTML*, que se ejecuta del lado del cliente.

- ***ASCII.*** (American Standard Code for Information Interchange; Código americano estándar para el intercambio de información). Código empleado para representar letras, números y otros símbolos en una computadora.

- ***ASP***. (Active Server Pages; Páginas activas de servidor) Tecnología propietaria de Microsoft para generar contenido dinámico en las páginas Web.

- ***Base de datos***. Conjunto de datos especialmente organizados para almacenar y recuperar la información de forma óptima.

- ***Bug***. Error en un programa de computadora que ocasiona un mal funcionamiento.

- ***Cliente***. Sistema de cómputo ó proceso que se encarga de realizar peticiones de servicio a otro sistema de cómputo o proceso.

- ***Compilador***. Programa que traduce código escrito en algún lenguaje de programación en instrucciones con otro formato (código objeto). Por ejemplo, un *compilador* de C traduce el código de C en instrucciones de lenguaje máquina.

- ***DBMS***. (DataBase Management System; Sistema manejador de bases de datos) Software que permite manipular la información en una base de datos, permitiendo almacenarla y recuperarla de manera eficiente y brindando un esquema de seguridad para proteger la información.

- ***DOS***. (Disk Operating System; Sistema operativo de disco) Sistema operativo empleado en computadoras personales. La versión más conocida es MS-DOS de Microsoft.

- ***GPL***. (General Public License; Licencia pública general). Licencia que se emplea con software de distribución libre. *GPL* permite a los usuarios copiar y modificar el software con la finalidad de mejorarlo o adaptarlo a necesidades propias, pero teniendo la obligación de ponerlo a disposición de toda la comunidad de Internet.

- ***Hipertexto***. documento que con un clic del ratón lleva al usuario a otro documento o alguna otra parte del mismo documento.

- ***HTML***. (HyperText Markup Language; Lenguaje de marcado de Hipertexto). Lenguaje utilizado para crear documentos que pueden ser vistos través de un navegador de Web.

- ***HTTP***. (HyperText Transfer Protocol; protocolo de transferencia de hipertexto) Protocolo de comunicación empleado para transferir documentos en el Web. *HTTP* define como deben ser formateados y transmitidos los mensajes y que acciones debe tomar tanto el servidor de Web como el navegador de páginas en respuesta a varias acciones.

- ***Intérprete***. Programa que traduce y ejecuta las instrucciones de un lenguaje de programación de alto nivel.

- ***Java***. Lenguaje de programación orientado al desarrollo de software multiplataforma y trabajo en red.

- ***JavaScript***. Lenguaje de programación cuyo código es interpretado en el navegador de Web. *JavaScript* se utiliza para diseñar páginas interactivas que con *HTML* es imposible realizar.

- ***JSP***. (*Java* Server Pages; Páginas *Java* en servidor). Tecnología que emplea el lenguaje de programación *Java*, para generar páginas de contenido dinámico. *JSP* es una alternativa a la tecnología de *ASP* de Microsoft.

- ***Linux***. Sistema operativo de distribución libre, multiusuario y multitareas. *Linux* está basado en *Unix* y está disponible para distintas plataformas de Hardware, que van desde computadoras personales hasta servidores de alto rendimiento.

- ***Normalización***. Proceso a través del cual se organiza la información de tal manera que se minimice la redundancia.

- ***PHP***. Lenguaje de programación que permite generar contenido dinámico. Esta tecnología funciona del lado del servidor, similar a *ASP* de Microsoft y *JSP* de Sun Microsystems.

- ***RXML***. (RoXen Markup Language; Lenguaje de marcado Roxen). Lenguaje con sintaxis similar a *HTML*, que se emplea del lado del servidor para generar páginas de contenido dinámico.

- ***RDBMS***. (Relational DataBase Management System; Sistema manejador de bases de datos). Es un DMBS que utiliza el enfoque relacional para manipular la información. El lenguaje estándar empleado para manipular información relacional es *SQL*.

- ***Servidor***. Dentro del esquema cliente/servidor, el servidor tiene la tarea de atender las peticiones de servicio que el cliente realiza.

- ***Servidor de Web***. Software que brinda el servicio de páginas de Internet utilizando *HTTP* como protocolo de comunicación. En ocasiones el término se emplea también para referirse al equipo de cómputo donde residen las páginas de Web

- ***Servlet***. Programa realizado en el lenguaje de programación *Java*, que se ejecuta del lado del servidor, pero a diferencia de los *JSP*, los *Servlets* deben de ser compilados.

- ***Sistema operativo***. Software de computadora que permite administrar los recursos de la computadora, como son: memoria, procesador y dispositivos de entrada y salida.

- ***Software de distribución libre***. Software, cuyo código fuente está disponible para toda la comunidad de Internet, con la finalidad de fomentar la participación en el perfeccionamiento del software. *GPL* es la licencia que está presente en este tipo de software.

- ***Software propietario***. Software cuyos detalles de diseño y funcionamiento son privados y únicamente son conocidos por la compañía que lo desarrollo. A diferencia del software de distribución libre, cualquier modificación a un *Software propietario* constituye un delito.

- ***SQL***. (Structured Query Language; Lenguaje de consulta estructurado) Lenguaje empleado para realizar manipulaciones en la información de una base de datos relacional. *SQL* es el lenguaje estándar para los *RDMBS*.

- ***Unix***. Sistema operativo, multiusuario y multitarea, desarrollado en la década de los 70's. *Unix* es frecuentemente ocupado para equipos que requieren de alto desempeño.

- ***VBScript***. Lenguaje desarrollado por la compañía Microsoft para competir con *JavaScript*. *VBScript* tiene una sintaxis similar a la de Visual Basic.

- ***XML***. (Extended Markup Language; Lenguaje extendible de marcado) Lenguaje empleado para definir contexto y atributos para el contenido de un documento.

- ***XHTML***. (Extended HyperText Markup Language; Lenguaje extendible de marcado para hipertexto). *XHTML* es esencialmente *HTML* modificado para ser compatible con *XML*.

B. Bibliografía.

B.1. Libros consultados.

- Chuck Musciano, Bill Kennedy. *HTML*. La guía completa. O'Reilly. 2ª. Edición. México 1999.
- Gold-Bernstein, Beth. Marca, David. Designing Enterprise Client/Server Systems. Prentice Hall. E.U. 1998.
- Cintron, Dave. Fast track Web Programming. Wiley. E.U. 1999.
- Spainhour, Stephen; Eckstein, Robert. WebMaster in a nutshell. 2ª. Edición. O'Reilly. E.U. 1999
- Date, C. J. Sistemas de bases de datos. Volumen 1. 5ª. Edición. Addison-Wesley. México, 1990.
- Pressman, Roger S. Ingeniería de software. Un enfoque práctico. 4ª Edición. McGraw Hill. México, 1998.

B.2. Direcciones de Internet consultadas.

- Comparación entre servidores para Internet.

http://www.serverwatch.com

- Página del proyecto de software libre GNU.

http://www.gnu.org

- Página del servidor de Web Apache Server.

http://www.apache.org

- Página del servidor de Web Roxen Server.

http://www.roxen.com

- Página oficial de *Linux* en México.

http://www.linux.org.mx

- Webopedia. Glosario de Computación

http://www.webopedia.com

- Oracle. *RDMBS* comercial.

http://www.oracle.com

- Sybase. Adaptive Server, *RDMBS* comercial.

http://www.sybase.com

- MySQL. *RDMBS* de distribución libre.

http://www.mysql.com

- *Java*. Lenguaje de programación multiplataforma.

http://www.javasoft.com

- *PHP*. Lenguaje de programación de distribución libre.

http://www.php.net