UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

MIGRACION DE LA BASE DE DATOS ACADEMICA DEL INSTITUTO DE INGENIERIA (BDAII) A INTERNET

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

YAZMIN HERNANDEZ GARCIA

DALILA PEREZ HERNANDEZ

DIRECTOR DE TESIS: ING. GABRIEL CASTILLO HERNANDEZ

MEXICO, D. F. NOVIEMBRE 1999

Dedico esta tesis a mi *madre*, por todo el cariño recibido desde mi infancia, por su permanente ejemplo de coraje y perseverancia. Agradezco todo su apoyo a lo largo de mi vida.

A *Manuel* y *Diana*, por haberme enseñado a fijarme metas y cumplirlas, con el compromiso que conlleva una labor tanto educativa como profesional.

A mi *padre*, gracias por todo lo que me ha dado.

A mis *abuelitos*, *tíos* y *primos* por todo el cariño brindado.

Por todo lo que ha existido entre nosotros, por lo que me has ayudado, por lo que me has enseñado y he aprendido de ti, gracias *Raúl*.

Al *Ing. Gabriel Castillo*, por ofrecerme todo su apoyo en la asesoría de esta tesis.

Alva y *Armando*: gracias por su amistad e invaluable ayuda.

Al *Ing. Palacios* y al *Ing. Ambriz* por darme la oportunidad de desarrollar los conocimientos adquiridos a lo largo de la carrera.

A mis *sinodales*, por el apoyo al leer esta tesis y por ofrecerme sus comentarios. Gracias por la formación que me han brindado dentro de la Universidad.

Araceli, *Margarita*, *Andrés* y *Rafael*: gracias por haber contado con ustedes en todo momento a lo largo de la carrera y por su amistad.

A mis amigos *Artemia*, *Elenita*, *las niñas*, *Alberto*, *Alejandro*, *David*, *Eduardo*, *Fernando*, *Guendaviani*, *Gustavo* les dedico las siguientes líneas: "Cuando Dios creó a los amigos, lo hizo pensando en que tuviéramos la posibilidad de elegir a un ser querido en el cual pudiéramos confiar. Alguien con quien podamos disfrutar nuestras aventuras, compartir nuestros fracasos. Pero al final de cada instancia recorrida no hubiera rivalidad sino un lazo de afecto que nos impulse a estar siempre juntos".

Yazmín Hernández García.

A *Dalila*:

Por su amistad, apoyo, comprensión y por haberme brindado la oportunidad de desarrollar esta tesis juntas, le dedico las siguientes líneas:

"Es difícil decir que es imposible. Los sueños de ayer son la esperanza de hoy y las realidades del mañana".

Muchas Gracias A:

Primero que nada quiero darle las gracias a *Dios* todo poderoso que siempre esta presente para ayudarme en todo momento.

A mis *padres* por enseñarme que cuando se quieren se pueden cumplir todas las metas que uno se propone.

Mamá gracias por estar conmigo hasta el día de hoy.

A mi hermana *Claudia* y a mi hermana *Mariana* por hacerme la vida más entretenida, sin sus problemas y disgustos mi vida sería demasiado monótona. Porque siempre me apoyaron y alentaron a seguir adelante.

A *Mario Enrique* porque con él estoy aprendiendo algo que no tuve oportunidad de vivir.

A *Yaz*, amiga y compañera, por su paciencia y su comprensión a lo largo de nuestra amistad.

A *Alva*, sin duda alguna por ser la persona que me enseñó lo que sé y me encaminó a la realización de esta meta.

A *Armando*, por ser la persona que confió en mi, por su amor, paciencia, comprensión y enseñarme que uno es tan grande como los sueños que se atreve a vivir. Por todos los detalles que me ha brindado durante este tiempo.

A *Raúl*, porque nunca me explicaba en lo que quería pero siempre me daba la respuesta correcta.

Ing. Gabriel, por el apoyo recibido, brindado aún en momentos difíciles y en especial por su paciencia y ayuda.

Al *Instituto de Ingeniería*, siempre estará en mi recuerdo, así como el *Ing. Luis Palacios Hammeken* y el Ing. *Marco Ambriz Maguey* por haberme permitido pertenecer al Instituto de Ingeniería.

Antes de acabar, quiero agradecer a todos los que han ayudado con su amistad, ayuda, constancia, apoyo y ánimos a que este proyecto vea la luz (*Alberto, Miguel, Guillermo, Las niñas, Elenita, Angélica, Araceli, Andrés y Rosario)* que siempre me han apoyado en las bajas y las altas a los cuales dejo con un pensamiento:

"Cuando te sientas que no eres nadie para el mundo, recuerda que eres el mundo para alguien".

Dalila Pérez Hernández

Horal

El mar se mide por olas
El cielo por alas
Nosotros por lágrimas

El aire descansa en las hojas
El agua en los ojos
Nosotros en nada

Parece que sales y soles
Nosotros y nada...

Jaime Sabines

ÍNDICE

INTRODUCCIÓN

Conocida como telaraña mundial, WWW, W3, o simplemente Web, constituye uno de los elementos más poderosos que sistematizan y simplifican el acceso a Internet. Este nuevo sistema ha revolucionado la forma en que los usuarios se comunican y utilizan los servicios de la llamada Red de Redes. El Web ha sido en parte la causa del espectacular aumento del número de usuarios de Internet y de la popularidad que ha adquirido la red en la actualidad, debido a la gran facilidad de manejo, que hace que el usuario "navegue" por WWW de forma muy sencilla y natural. El sistema WWW presenta la información y el acceso a los distintos recursos a través de documentos de hypertexto, también llamados páginas Web.

El WWW ha contribuido de manera decisiva a que personas que no estaban inmersas en el mundo de la informática se acerquen a este medio de comunicación: Internet. Los servidores WWW distribuidos a través de todo el mundo dan forma a esta inmensa red de computadoras que se conectan a través de los documentos de hipertexto.

Para lograr que los servidores WWW se conecten y obtener la información que brindan es necesario disponer de un programa que se encarga de realizar dicha conexión. A estos servidores se les conoce como browsers o navegadores. Los más conocidos en el medio son el Internet Explorer de Microsoft y el Netscape Navigator, de Netscape. En la actualidad, son muchas las versiones de navegadores existentes en el mercado, e incluyen cada vez más servicios que permiten utilizar las otras herramientas de Internet.

Los sistemas de información son fundamentales para las distintas organizaciones ya que la información es uno de los recursos más valiosos de cualquier institución o empresa. Casi todas las organizaciones tienen cantidades masivas de información que necesitan ser clasificadas, almacenadas y consultadas continuamente. El uso de las tecnologías de Internet dentro de redes corporativas, permite a los usuarios el acceso a la información en forma sencilla, rápida y eficiente.

Con todas las ventajas que ofrece el WWW, se pensó en migrar la *Base de Datos Académica del Instituto de Ingeniería (BDAII)* al Web. Este sistema permitirá de manera rápida, fácil y segura ingresar información a su curriculum vitae o simplemente consultarla desde cualquier parte donde se cuente con Internet.

El sistema de la *Base de Datos Académica del Instituto de Ingeniería (BDAII)* está basado en una arquitectura Cliente/Servidor, en la parte del cliente se tiene una interfaz para actualizar, ingresar o consultar la información de un curriculum, por otra parte el servidor es Sybase como sistema manejador de bases de datos relacionales(RBDMS) y el servidor Web Apache HTTPD.

La interfaz del cliente en Web de la *Base de Datos Académica del Instituto de Ingeniería (BDAII,* esta integrada por varios rubros, como son: Altas. Bajas, Cambios, Consultas, Semblanza, Curriculum General, Ayuda, así como la opción de Comentarios la cual servirá para recibir las sugerencias de los usuarios.

Dentro de los beneficios que se obtendrán al migrar este sistema son:

- Que los usuarios consulten de manera rápida y eficaz su información curricular desde el Web del Instituto de Ingeniería.

- Evitar instalaciones del antiguo cliente desarrollado en VisualBasic, prescindiendo así el uso de recursos, dinero y tiempo.

- Agilizar la tarea de captura o actualización de la información curricular ya que cada quien será responsable de la misma.

Está tesis habla sobre los trabajos realizados para el desarrollo del sistema de la *Base de Datos Académica del Instituto de Ingeniería (BDAII)* en el WEB.

El presente documento consta de tres capítulos. El capítulo uno describe al Instituto de Ingeniería y su problemática, así como las soluciones que se le han dado al problema. En el capítulo dos presentamos la base de datos ya existente, así como un análisis de información a través de un diagrama de flujo de información, las fuentes que existen y hacia donde van. En el capítulo tres describe el diseño en Web de las páginas de la *Base de Datos Académica del Instituto de Ingeniería (BDAII),* así como un mapa de navegación con un mecanismo de integración y recursos utilizados según las necesidades de los usuarios. Se explican las validaciones que se consideraron importantes para evitar ingresar información errónea a la base de datos. Se cuenta con apéndices sobre las herramientas y lenguajes utilizadas en la realización del presente documento. Por último se presentan los comentarios finales originados por el desarrollo del sistema y se describen tendencias en los sistemas de información.

Con este trabajo se pretende que el sistema de la *Base de Datos Académica del Instituto de Ingeniería (BDAII)* en WEB proporcione servicios de manera factible, que refleje una estructura organizada al agrupar los procesos realizados, así como también eliminar los procedimientos de instalación de la versión VB y difundir la información de los investigadores del Instituto de Ingeniería.

CAPÍTULO I
EL PROBLEMA DE LA INFORMACIÓN DEL INSTITUTO DE INGENIERÍA

1.1 Introducción.

El Instituto de Ingeniería de la Universidad Nacional Autónoma de México es el centro de investigación en diversas áreas de la ingeniería más productivo del país. Es una comunidad de aproximadamente 900 personas, a saber: investigadores, estudiantes de ingeniería que realizan trabajos de tesis de licenciatura, maestría y doctorado, técnicos académicos, personal secretarial y de servicios. Sus instalaciones ocupan 12 edificios en la zona de Ciudad Universitaria, en Coyoacán, con una extensión de 63,000 metros cuadrados entre laboratorios, cubículos, áreas comunes y un auditorio.

Desde su fundación, la política del Instituto ha sido realizar investigación orientada a problemas generales de la ingeniería, así como colaborar con entidades públicas y privadas para mejorar la práctica de la ingeniería en el ámbito nacional, al aplicar los resultados de las investigaciones a problemas específicos.

Como consecuencia, algunos proyectos son financiados con recursos que la UNAM otorga, y otros, mediante contratos de investigación con empresas o corporaciones solicitantes.

1.1.1 Historia

En 1936 se crearon los laboratorios de ingeniería experimental en la Comisión Nacional de Irrigación, para la aplicación de métodos experimentales a la solución de problemas de ingeniería civil, los cuales a su vez dieron origen a la fundación del Instituto de Ingeniería como asociación civil en al año de 1956. En 1957, por gestiones del doctor Nabor Carrillo y del ingeniero Javier Barros Sierra, respectivamente rector de la Universidad y director de la Escuela Nacional de Ingeniería, el Instituto pasó a ser la División de Investigación de la Escuela (hoy Facultad) hasta que, el 27 de julio de 1976, el Instituto de Ingeniería se constituyó oficialmente como dependencia universitaria, por acuerdo del Consejo Universitario.

1.1.2 Finalidad y orientación.

El Instituto de Ingeniería es parte del Subsistema de Investigación Científica de la Universidad Nacional Autónoma de México y orgánicamente se encuentra dentro de la Coordinación de la Investigación Científica.

Las principales funciones del Instituto son desarrollar la investigación para mejorar los conocimientos, métodos y criterios en ingeniería, contribuir a la formación de expertos en esta rama del saber, así como promover la más alta calidad en la práctica profesional.

En los programas de trabajo se enfatiza el interés en las necesidades de la ingeniería nacional actuales y previsibles. Las actividades que se llevan a cabo en el Instituto son: Investigación técnica y aplicada, apoyo al desarrollo tecnológico y análisis de los requerimientos sociales a cuya solución puede aportar la ingeniería. Asimismo, se proporcionan servicios de ingeniería a los diversos sectores de la sociedad con el propósito de contribuir al avance de los objetivos propio de la universidad.

1.1.3 Funciones

Las principales funciones del Instituto son:

- Realizar investigación para mejorar los conocimientos, métodos y criterios en ingeniería, tanto fundamental como aplicada. Formación de investigadores en ingeniería.
- Apoyar al desarrollo tecnológico y análisis de los requerimientos sociales a cuya solución puede aportar la ingeniería.
- Proporcionar servicios de ingeniería a los diversos sectores de la sociedad con el propósito de contribuir al avance de los objetivos propios de la Universidad.
- Apoyar la formación de profesores y las tareas docentes de la Facultad de Ingeniería.
- Estudiar problemas de interés nacional.
- Difundir los resultados de sus investigaciones.
- Llevar a cabo las actividades necesarias para realizar las funciones precedentes.
- En el desempeño de estas funciones, el Instituto colabora con otras instituciones afines, técnicas, culturales y científicas, del país y en el extranjero.

1.1.4 Política.

La política fundamental del Instituto, desde su fundación en 1956, ha sido ocuparse de la investigación orientada a problemas generales de la ingeniería cuya importancia es mundial, y de apoyar a la vez a las instituciones privadas públicas para mejorar la práctica de la ingeniería en México. Se trata así de aplicar los resultados de las investigaciones propias a problemas específicos del país.

Como consecuencia, algunos proyectos del Instituto son financiados con recursos que la propia Universidad otorga, y otros, con aportaciones que resultan de contratos con empresas o instituciones interesadas en algún resultado específico.

En congruencia con estas actividades, el Instituto confiere relevancia especial a su tarea de formación de personal especializado, para lo cual incorpora en sus proyectos numerosos estudiantes de licenciatura y posgrado. Estos estudiantes se benefician al completar su formación con la práctica de la investigación en ingeniería, y contribuyen a los resultados de los proyectos. El Instituto cuenta con un programa propio de becas que complementa otros programas vigentes en la Universidad y que permite enfatizar la importancia de la actividad docente.

1.2 La información curricular del personal académico en el Instituto de Ingeniería.

En el Instituto de Ingeniería se cuenta con una gran cantidad de información, la cual debe ser analizada y estructurada de acuerdo a las necesidades del personal.

1.2.1 Problema.

En el Instituto de Ingeniería es necesario disponer de información tanto administrativa como académica. Para la primera existen sistemas que se encargan de organizarla y presentarla en forma adecuada. En la segunda, la información disponible es limitada, y en general está íntimamente relacionada con las necesidades administrativas. Disponer de un sistema capaz de ordenar y presentar adecuadamente información académica referente a publicaciones, proyectos, convenios, etc., es fundamental para una adecuada visión de las tareas académicas del Instituto de Ingeniería. Con todo ello en mente se diseñó e implantó una base de datos denominada *Base de Datos Académica del Instituto de Ingeniería (BDAII).*

Para la alimentación de la BDAII se emplean los currículos del personal académico, buscando en todo momento evitar duplicidad e inconsistencias. De esta manera la información se mantiene actualizada, pues el académico es responsable del adecuado mantenimiento de su currículum. Con ello se tendrán ventajas de confiabilidad y oportunidad, pues el mismo personal tiene interés en mantenerlo actualizado y será el medio que permita reportar tanto ventajas como desventajas de su uso. El formato que el sistema presente debe ser útil para el académico, pero congruente con las necesidades requeridas por el Instituto de Ingeniería y otras instancias relacionadas.

1.2.2 Necesidad.

El Instituto de Ingeniería se caracteriza por tener personal que dentro de las distintas ramas de ingeniería, se destaca en el ámbito científico, industrial y de docencia. A través del tiempo solo existen registros aislados de la aportación que hace el Instituto a la sociedad y no se tiene algún registro histórico de lo que se ha realizado al paso de los años. Con el fin de establecer las necesidades primordiales se ha establecido que el sistema debe servir como base para:

- La evaluación del personal del Instituto de Ingeniería.
- Contener información del personal del Instituto como:
  - La formación académica del personal.
  - Los cargos desempeñados del personal.
  - Participación en la formación de recursos humanos.
  - Distinciones recibidas.

- Contener información de productos que fueron creados por el cuerpo de investigadores:
  - Tesis dirigidas.
  - Informes de proyectos.
  - Producción científica: Artículos en revistas, memorias en congresos, informes de proyectos, normas, patentes, libros escritos.
- Difundir a través de un medio electrónico tal como Internet (WWW) cierta información como la producción científica y las tesis.

1.2.3 Objetivo.

La Base de Datos Académica del Instituto de Ingeniería (BDAII), contará con un registro histórico de las investigaciones realizadas, proyectos desarrollados y la producción escrita del personal académico. Además tiene el propósito de manejar en su totalidad la información de los currículos del personal del Instituto de Ingeniería, de manera que los datos requeridos para cualquier evaluación, particularmente las del Instituto de Ingeniería, se puedan consultar con facilidad.

1.2.4 Puntos considerados en el desarrollo de la Base de Datos Académica del Instituto de Ingeniería.

El diseño de un sistema de información, es una tarea que involucra tanto al personal de sistemas de cómputo como las personas que podrán explotar este sistema. Por lo tanto es necesario establecer una metodología para definir las necesidades del personal involucrado en este sistema.

La información básica que se requiere para el sistema se basa fundamentalmente en los datos que el investigador tiene en su currículum, sin embargo si el sistema tiene como fuente los currículos, estos pueden presentar una desventaja a cuidar: la duplicidad. Es decir, los proyectos son realizados por varias personas, por lo tanto cuando se capture la información se puede duplicar ésta en la base de datos.

Es importante también definir las necesidades primordiales de información que contendrá la BDAII. A partir de un análisis detallado se determinó que la información en la BDAII estará organizada de acuerdo con los siguientes rubros:

- Producción académica: Libros, artículos en revistas, memorias.
- Vinculación con la sociedad: Proyectos realizados, asesorías, consultoría externa.
- Docencia: Tesis dirigidas, asesorías, conferencia, cursos impartidos.
- Formación académica: Estudios profesionales, congresos asistidos, estancias de investigación.
- Premios y distinciones.
- Nombramientos.

Además debe proporcionar elementos suficientes para:

- Conformar una base histórica de las investigaciones realizadas en el Instituto de Ingeniería.
- Servir como fuente a otras evaluaciones.
- Difusión nacional e internacional de proyectos realizados y tesis dirigidas en el Instituto de Ingeniería para el caso de que se despliegue la información en WEB del Instituto.
- Suministrar información general del personal académico así como la impresión de sus currículos respectivos.
- Realizar estadísticas internas utilizadas por órganos tales como el Consejo Interno y estadísticas externas para UNAM, DGAPA, DGESII, DGIA.

La información que tendrá la base de datos se estructurará en los siguientes rubros:

- Datos personales.
- Formación académica:
  - Grados obtenidos
  - Otros estudios
  - Congresos y similares
  - Estancias
  - Idiomas.
- Cargos desempeñados.
- Actividades docentes:
  - Cátedras
  - Otros cursos
  - Conferencias impartidas
  - Tesis dirigidas
  - Tutorías.
- Distinciones.
- Publicaciones:
  - Libros y capítulos
  - Artículos en publicaciones periódicas
  - Informes
  - Memorias
  - Otras publicaciones.
- Desarrollos:
  - Patentes
  - Otros.
- Semblanza.

Es importante establecer que quiénes utilizarán el sistema de información, por una parte será cada investigador tal vez, a través de la secretaria correspondiente y por otra será la Secretaría Académica, para realizar las evaluaciones correspondientes.

1.2.5 Impacto del sistema.

El desarrollo de la BDAII ofrecerá los siguientes beneficios a los usuarios:

- Se disminuirá la cantidad de informes para evaluaciones durante el año.
- Consultarán información a través de la computadora.
- Difusión de su trabajo en otros niveles internacionales, a través del Web, entre los posibles catálogos está: Tesis, artículos, libros, proyectos.
- Oportunidad de imprimir cierta información o de tenerla en algún medio electrónico para poder adecuarla a sus necesidades.

1.3 Sistema de captura y almacenamiento de la información curricular.

Como resultado del análisis anterior, en 1994 se decidió crear un sistema que cumpliera con los requisitos para cubrir funcionalmente las necesidades de estructurar la información:

1.3.1 Solución Visual Basic

Actualmente el investigador cuenta con una base de datos que tiene como cliente un sistema desarrollado en Visual Basic. Esta forma de capturar la información no ha sido explotada debidamente por cada investigador, ya que todo la captura de su información curricular se hace a través de cada secretaria la cual se encarga de presentarle al investigador la información en el formato final de su curriculum, el investigador hace las correcciones necesarias y se las devuelve a la secretaria para su debida actualización.

Cuando el investigador desea actualizar su curriculum, también debe hacerlo a través de la secretaria. Esta mecánica hace que este sistema sea poco atractivo para los investigadores, además; para que cada investigador cuente con la instalación del cliente BDAII debe cumplir con los siguientes requisitos:

Una máquina 486, con 20 MB de espacio libre en disco duro, sistema operativo Windows 95 ó 98, así como disponibilidad de horario de mínimo una hora para que se realice la instalación en la fecha y hora fijada. Además se tiene un problema con el servidor de bases de datos, debido a que solo tiene 64 licencias, y cada instalación en Visual Basic ocupa cinco conexiones, eso nos limita a doce usuarios; actualmente se cuenta con aproximadamente 200 investigadores y eso pensando que solo esta aplicación estuviera funcionando, ya que existen otras aplicaciones montadas sobre el servidor de base de datos.

1.3.2 Solución Web

Para hacer más atractivo el sistema para los investigadores se decidió migrar esta aplicación al Web, lo cual resulta más eficiente para los investigadores ya que desde una dirección electrónica en especial se conectan a la aplicación, por medio de una contraseña entran

directamente a su información. Ciertamente la aplicación en VB era útil, por lo que, se decidió usar esta aplicación como un modelo base para el desarrollo del sistema en Web.

1.4 Recursos.

Los recursos necesarios para el sistema son:

- Un manejador de base de datos para la administración de la información:
  SYBASE[1], en particular el producto Adaptive Server Enterprise con 64 licencias.
- Una herramienta para desarrollar el cliente del sistema en Web:
  Se seleccionó a Web.sql en su versión 1.1.
- Un servidor de Web:
  Apache en su versión 1.2.6.
- Una estación de trabajo donde resida el servidor de SYBASE:
  Sé hizo la selección del equipo existente, para este caso fue una Sparc Station 20 que utilizará un espacio en disco de aproximadamente 5 GB, memoria de 128 MB.
- El sistema necesitará como mínimo computadoras con las siguientes características:
  Desde una PC 80386 con 16 MB en RAM y con un espacio de disco duro de 500 MB que contenga algún tipo de visualizador como Netscape o Internet Explorer.

1.5 Desarrollo.

En la siguiente figura se muestra como se encuentra integrado la arquitectura general del sistema BDAII Web.

Arquitectura general del sistema BDAII Web

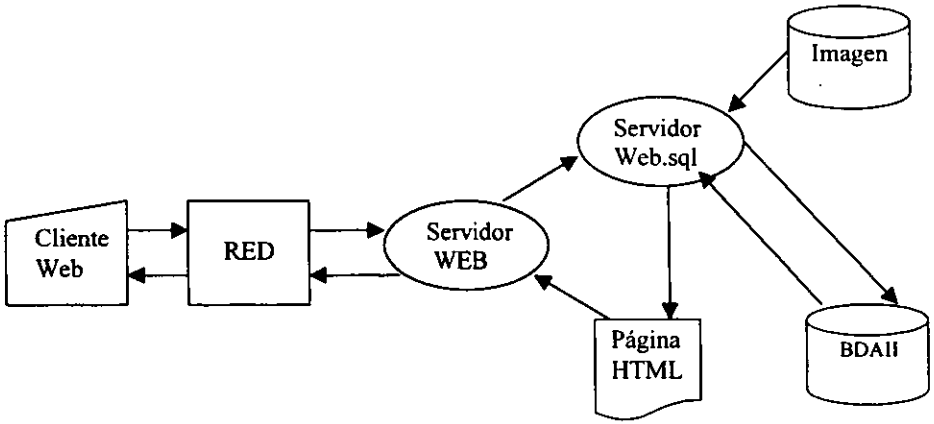


Fig.1.1 Representa la arquitectura general de la aplicación de BDAII Web

[1] Para mayor información consultar el apéndice en el inciso G.

A continuación se describirá el funcionamiento de la arquitectura presentada en la figura anterior:

En el momento en que el usuario consulta el Web del Instituto de Ingeniería por medio de un browser (Netscape, Internet Explorer u cualquier otro), este browser realiza una petición al servidor Web a través de un Uniform Resource Location (URL). Este URL es interpretado por el servidor HTTP dentro de la ruta de un archivo en el servidor. Si el archivo tiene un formato .html, .gif, .jpeg, u otro tipo de formato que el servidor Web reconozca, el servidor HTTP envía el archivo directamente al browser.

Sin embargo, cuando un browser solicita un URL que es interpretado por un archivo .hts (HyperText Sybase) o por un archivo .pl de Perl (solo Perl, no HTML), el servidor HTTP envía la petición al programa de Web.sql.

El programa de Web.sql lee los archivos .hts o .pl, procesa las peticiones de la base de datos, así como el código de Perl contenidos en el archivo utilizando un mapa de acceso para determinar qué base de datos se va a utilizar. El programa Web.sql compone la salida en el formato estándar HTML para el servidor HTTP y éste finalmente lo envía al browser para desplegarlo al usuario.

1.6 BDAII en Web y herramientas de desarrollo.

El Instituto de Ingeniería requiere contar con un sistema, a través de la cual sus investigadores de las distintas ramas de ingeniería puedan dar de alta sus registros, actualizar, consultar, eliminar o simplemente imprimir de una forma rápida y fácil su currículum completo o la parte que necesiten.

Por lo anterior es necesario desarrollar una aplicación, la cual no sea necesario instalarla en cada máquina del investigador que la solicite, condicionándolo a que necesita contar con ciertos requerimientos de hardware y software. El sistema podrá ser consultado vía Web desde el lugar que se desee y que además permita que el número de investigadores que se puedan conectar en cierto momento sea ilimitado.

Se desarrolló un sistema similar al ya existente pero utilizando páginas HTML y apoyado en Apache HTTP Server, JavaScript, Perl, Transact-SQL, Web.sql con la finalidad de que la consulta y actualización de la información pueda realizarse desde cualquier browser (Netscape, Internet Explorer).

El nuevo sistema difiere en algunos aspectos respecto al ya existente en VB, principalmente debido a la tecnología que se empleó en su desarrollo.

Las pantallas desarrolladas en el sistema actual cambiaron con respecto a las de Visual Basic en forma y comportamiento, debido, principalmente, a la diferencia de filosofía

empleada, sin embargo se buscó en todo momento la mayor similitud posible entre ambos sistemas, con el fin de que el nuevo sistema tenga amplia aceptación por parte de los usuarios actuales. Cabe mencionar que se implementó el módulo de bajas por separado, ya que originalmente en la interfaz Visual Basic se encontraba junto con el de Cambios.

Se utilizaron los siguientes productos para el desarrollo del sistema antes propuesto:

HTML, es un lenguaje muy sencillo que permitió describir hipertexto es decir, texto presentado de forma estructurada y agradable. La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, citas, etc.) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita o un gráfico determinado) y posteriormente la presentación final de dicho hipertexto la realiza un programa especializado (Netscape, Internet Explorer, etc.).

Apache HTTP Server, es mantenido por el grupo Apache y es derivado del original NCSA HTTPD 1.3. Gracias a él podemos practicar la creación y publicación de documentos HTML de la misma forma que se hace en Internet con una estabilidad y eficacia ampliamente comprobada en la gran cantidad de servidores apache actualmente en uso.

JavaScript, es un lenguaje de desarrollo de aplicaciones Cliente/Servidor a través de Internet. El programa en JavaScript tiene la particularidad de que esta contenido dentro del mismo documento HTML que lo presenta al usuario y por lo tanto no es un programa aparte. Permite crear aplicaciones similares a los CGIs. El programa en JavaScript reconoce eventos creados por el usuario, definiendo así un sistema interactivo. Podemos por ello crear formularios que verifiquen la validez de la información e interpreten esta en el mismo programa que interpreta el documento HTML, sin necesidad de comunicación por la red. El lenguaje JavaScript es un subconjunto del lenguaje Java, su mayor limitante es la de tener un conjunto de clases predefinidas sin posibilidad de crear nuevas.

Al implementar la validación del login y password se utilizó JavaScript, para crear la caja en la cual se pide sea suministrado el login y password del usuario, manipulando con código para enviar mensajes de error cuando dicho login o password sea incorrecto, además de que se utilizó para validar información en cada una de las formas presentadas al usuario y para el manejo de eventos en cada uno de los botones que son presentados en dichas formas.

Al implementar la parte de validación de algunos formularios así como encriptación del password se utilizó **Perl**, el cual es un lenguaje interpretado, que facilita la manipulación de procesos. Debido a que es un lenguaje interpretado y no compilado, se hace referencia a programas como scripts, siendo un script un programa interpretado al momento de

ejecutarse, reservando el nombre de programas para aquellos desarrollados en lenguajes compilados. Es una utilería que facilita el proceso de grandes volúmenes de información sin limitar su rendimiento.

Transact-SQL, SQL (Structured Query Language - Lenguaje estructurado de consultas) es un lenguaje de alto nivel para sistemas de bases de datos relacionales. Desarrollado originalmente por el laboratorio de investigación de IBM en San José a finales de la década de los años70's, SQL ha sido adoptado y adaptado en muchos sistemas de administración de bases de datos relacionales.

Aunque la "Q" de SQL significa "Query" (consulta), SQL incluye comandos no solo para la consulta (recuperación de datos) de una base de datos, sino también para la creación de bases de datos y objetos de bases de datos, adición de datos nuevos, modificación de datos existentes y otras funciones.

Sybase **Web.sql**, es una herramienta para Internet e Intranets con una nueva tecnología que facilita el acceso a bases de datos relacionales desde el World Wide Web, así como también la creación dinámica de documentos personalizados en formato HTML. El resultado es un mejor rendimiento en el acceso a la base de datos y en tiempo de respuesta. Además, Web.sql integra la tecnología Open Client de Sybase, lo cual permite que los datos de cualquier fuente sean incorporados dinámicamente en las páginas Web. Con Web.sql se extienden las funciones de un servidor Web posibilitando la inserción de instrucciones de bases de datos, tales como sentencias SQL, así como también códigos de escritos en Perl, dentro de un documento en formato HTML.

La interfaz Web.sql reconoce dos tipos de extensiones de archivos: .hts (Hyper Text Sybase) y .pl (Perl). Un script Perl tiene la facultad de interactuar con una base de datos, tal como lo hace un archivo HTS a través de sentencias SQL.

En los siguientes capítulos se presenta a la BDAII Web como una aplicación donde todas estas herramientas se han conjuntado obteniéndose un sistema práctico y fácil de operar.

CAPÍTULO 2
REQUERIMIENTOS DEL SISTEMA EN WEB A PARTIR DEL SISTEMA VB.

Una vez estudiadas cada una de las herramientas de desarrollo, definido el problema y el objetivo de la BDAII Web, se procedió a realizar un análisis del funcionamiento del sistema en VB, con ello se estableció la secuencia de "viaje" dentro de WEB y por lo tanto la funcionalidad esperada del sistema BDAII Web.

2.1 Esquema de navegación Visual Basic

El sistema en Visual Basic está formado de los módulos siguientes:

- Altas
- Cambios
- Consultas
- Ayuda
- Impresión
- Salir

La información que se almacena en la Base de Datos Académica, está organizada en los siguientes rubros:

- Datos personales
- Formación académica:
  - Grados obtenidos
  - Otros estudios
  - Congresos y similares
  - Estancias
  - Idiomas
- Cargos desempeñados
- Actividades docentes:
  - Cátedras
  - Otros cursos
  - Conferencias
  - Tesis dirigidas
  - Tutorías
- Distinciones
- Publicaciones:
  - Libros y capítulos
  - Artículos en publicaciones periódicas
  - Informes
  - Memorias
  - Otras publicaciones
- Desarrollos:
  - Patentes

  - Otros
- Semblanza

El sistema permite realizar altas, cambios y consultas de la información antes mencionada. Además se cuenta con un mecanismo de ayuda donde se visualiza la siguiente información:

- Como empezar
- Descripción del sistema.
- Índice.
- Créditos del sistema.

A continuación se presentarán algunas pantallas que muestran los módulos del sistema desarrollado en Visual Basic.

La Figura 1, presenta todos los módulos con los que cuenta el sistema y en especial la selección del módulo de Altas en el rubro grados obtenidos.

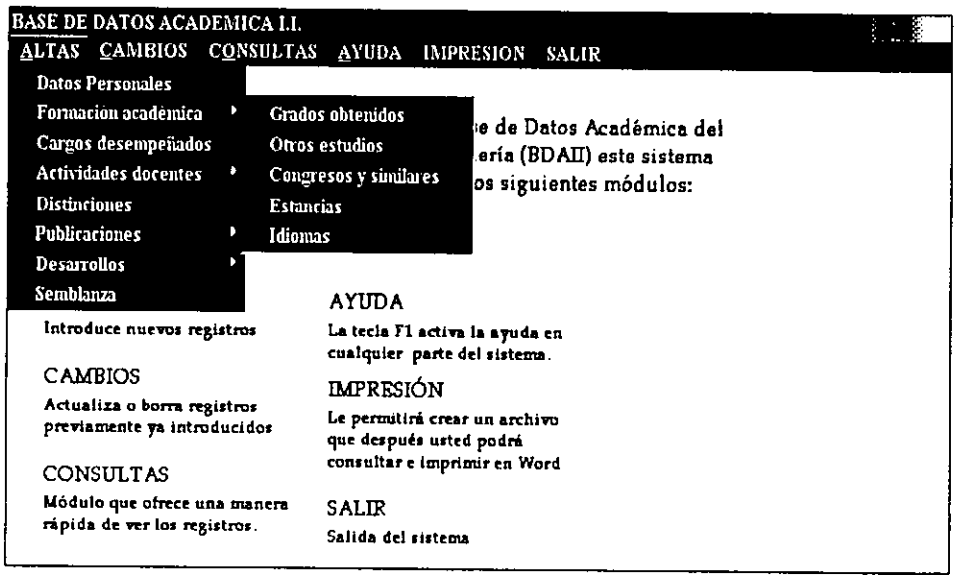


Figura 1: Presentación del menú principal para realizar una alta en el rubro grados obtenidos

En la Figura 2, se observa toda la información que se solicita para realizar una alta en dicho rubro, los datos deben ser suministrados por el usuario con el fin de que sean veridicos.

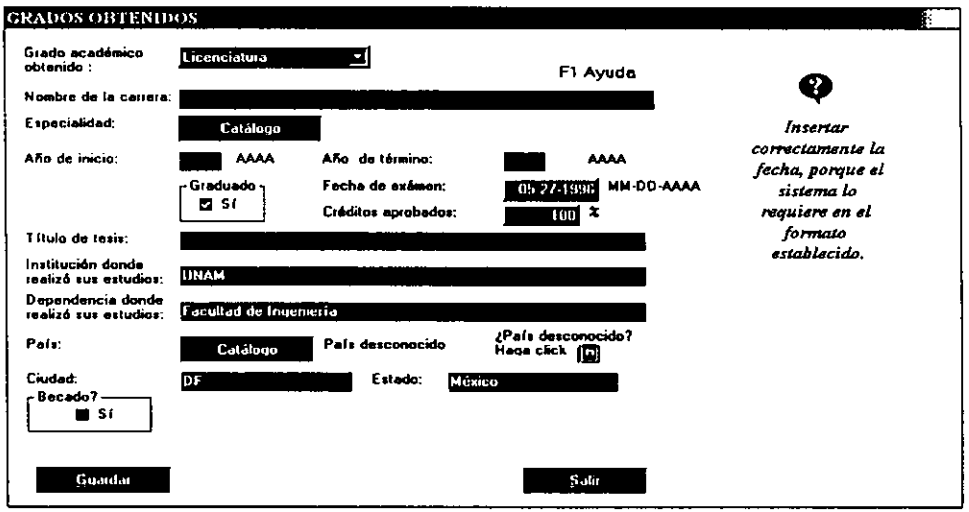


Figura 2: Información solicitada para realizar una alta en el rubro grados obtenidos.

En la Figura 3, se presenta nuevamente el menú principal pero con la selección del módulo de cambios o bajas.

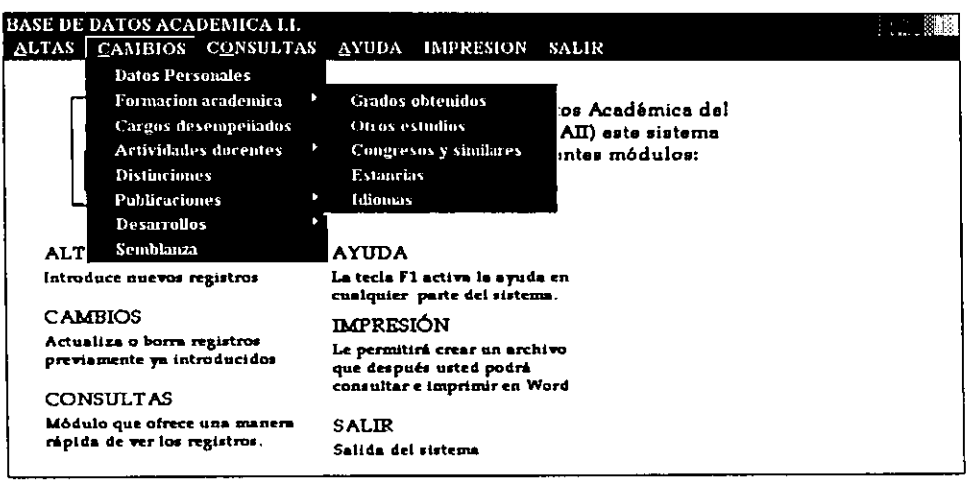


Figura 3: Presentación del menú principal para realizar un cambio o baja en el rubro grados obtenidos

La Figura 4, es la pantalla que se muestra para que el usuario seleccione el registro que desea cambiar o dar de baja.

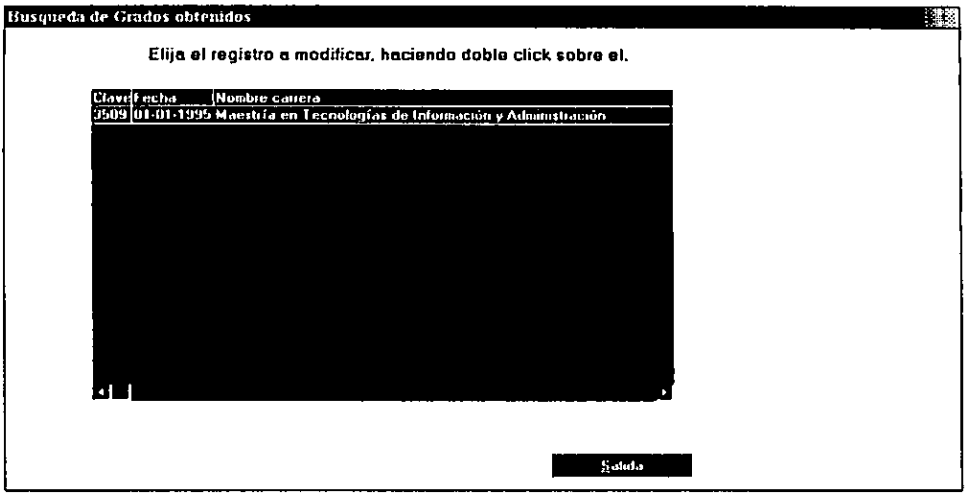


Figura 4: Elección del registro a cambiar o dar de baja en el rubro de grados obtenidos.

En la Figura 5, se observan los datos presentados al usuario con el fin de tener la posibilidad de cambiar algunos datos que aparecen en dicho registro o en su defecto darlo de baja, eliminando así toda la información que este contenga.

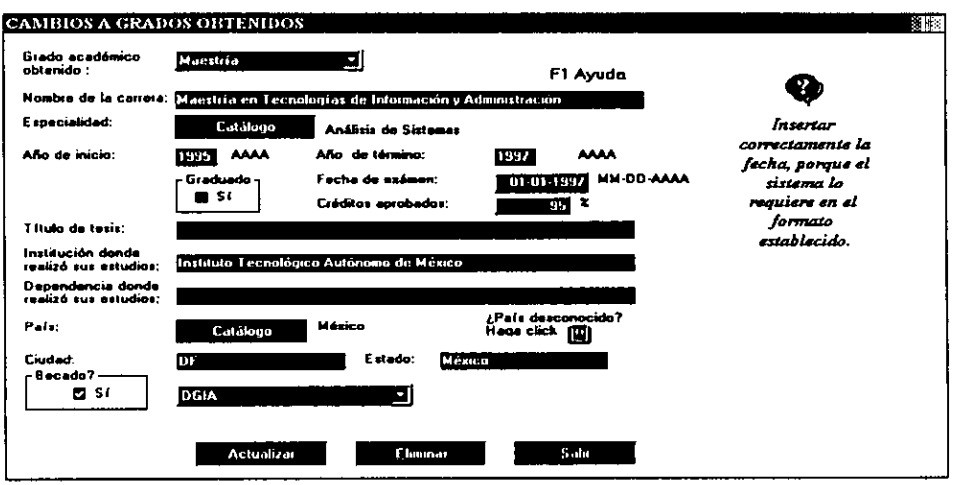


Figura 5: Cambio o eliminación de un registro en el rubro grados obtenidos.

2.2 Esquema de navegación en Web

El sistema BDAII en Internet consistirá de los siguientes módulos:

- Altas
- Bajas
- Cambios
- Consultas
- Semblanza
- Currículum General
- Ayuda
- Comentarios
- Salir

- Módulo Altas: El módulo de Altas permite la inserción de registros en la Base de Datos Académica del Instituto de Ingeniería, validando previamente la información suministrada por el usuario con el fin de que los datos almacenados sean correctos.

- Módulo Bajas: El módulo de Bajas permite eliminar los registros en la Base de Datos Académica del Instituto de Ingeniería, preguntando previamente si se desea en realidad eliminar la información que ha sido suministrada por el usuario. Después de haberse dado de baja no podrá recuperarse la información, por lo que es de vital importancia que antes de hacerlo se esté seguro que no desea contar con dicha información.

- Módulo Cambios: El módulo de Cambios permite actualizar los datos ya existentes en el sistema de los registros en la Base de Datos Académica del Instituto de Ingeniería, en aquellos rubros que se consideren prudentes, validando previamente la información suministrada por el usuario con el fin de que los datos almacenados sean correctos.

- Módulo Consultas: El módulo de Consultas permite visualizar la información de los registros en la Base de Datos Académica del Instituto de Ingeniería.

- Opción Semblanza: Permite consultar la línea de investigación del académico en los últimos años, además de todos aquellos aspectos que se consideran sobresalientes y que no se contemplaron en ninguno de los apartados anteriores.

- Opción Curriculum General: En este apartado se visualiza toda la información que ha sido capturado a través de los diversos rubros.

- Módulo Ayuda: Este apartado tiene como objetivo servir de guía en caso de tener problemas con el manejo del sistema, o bien problemas relacionados con el llenado correcto de datos.

- Opción Comentarios: En este apartado se pueden enviar, vía correo electrónico, los comentarios que se consideren prudentes sobre el sistema, o bien sobre aquellos en los que se tenga dudas aún después de haber consultado la ayuda.

- Opción Salir: Permite salir del sistema.

La BDAII Web incluirá por supuesto todos los rubros con los que contaba el sistema anterior.

A fin de presentar el esquema general de altas, bajas y cambios, hablaremos en particular del rubro de grados obtenidos en la aplicación desarrollada para el Web.

Para dar de alta un registro (Figura 6), en la página de grados obtenidos, hay que accesar el módulo de altas y posteriormente seleccionar dicha página del menú de formación académica.

La primera página, permite dar de alta información referente a los títulos logrados en estudios universitarios de grado, es decir: **licenciatura, maestría y doctorado**.

Nombre de la carrera: Es el nombre oficial que aparece en el título.

Especialidad: Es la especialidad sobre la que se efectuó el título.

La segunda página permite dar de alta información referente a:

Año de inicio: Se refiere al año en el que se comienzan los estudios sobre la carrera o grado.

Año de término: Se refiere al año en que se cursó la última materia de esa carrera o grado.

Fecha de examen: Es la fecha en la que presentó el examen profesional.

Graduado: Aquí se indica si la persona está graduada o no. Si no se está graduado, se pasa al inciso donde se solicita el porcentaje de créditos obtenidos; si ya se está graduado, además de marcar afirmativo este punto, se escribe el título de la tesis.

Créditos obtenidos: Para obtener el porcentaje de créditos obtenidos se suman los créditos correspondientes a las materias aprobadas.

Título de tesis: Es el nombre que aparece en la impresión de la misma.

Institución donde realizó sus estudios: Es el nombre de la universidad donde se realizaron los estudios.

Dependencia donde realizó sus estudios: Es el nombre de la dependencia donde se obtuvo el grado.

País: Es el nombre del país donde se obtuvo el grado.

Ciudad: Es el nombre de la ciudad correspondiente.

Estado: Estado donde se ubica esa institución.

Becado: Se elige este cuadro si se tuvo una ayuda económica durante los estudios para lograr el grado que se está registrando.

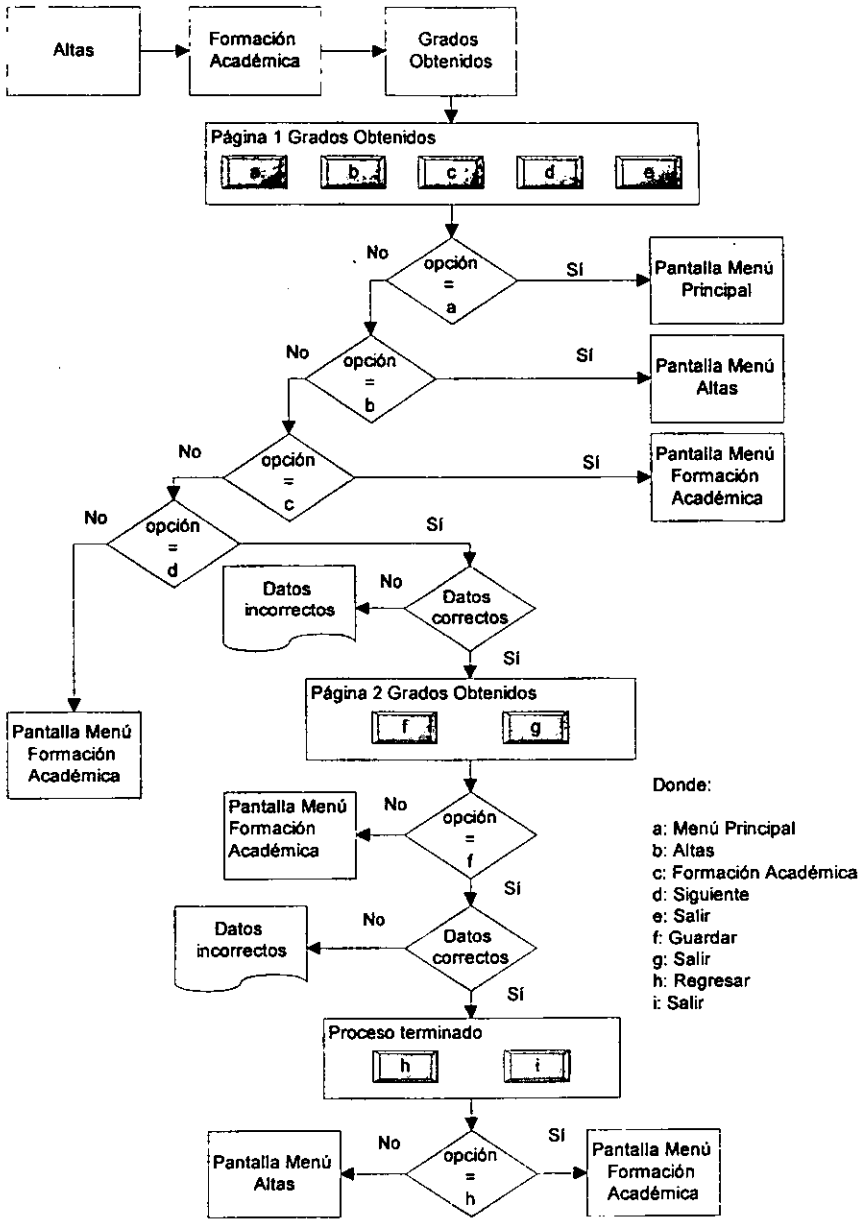
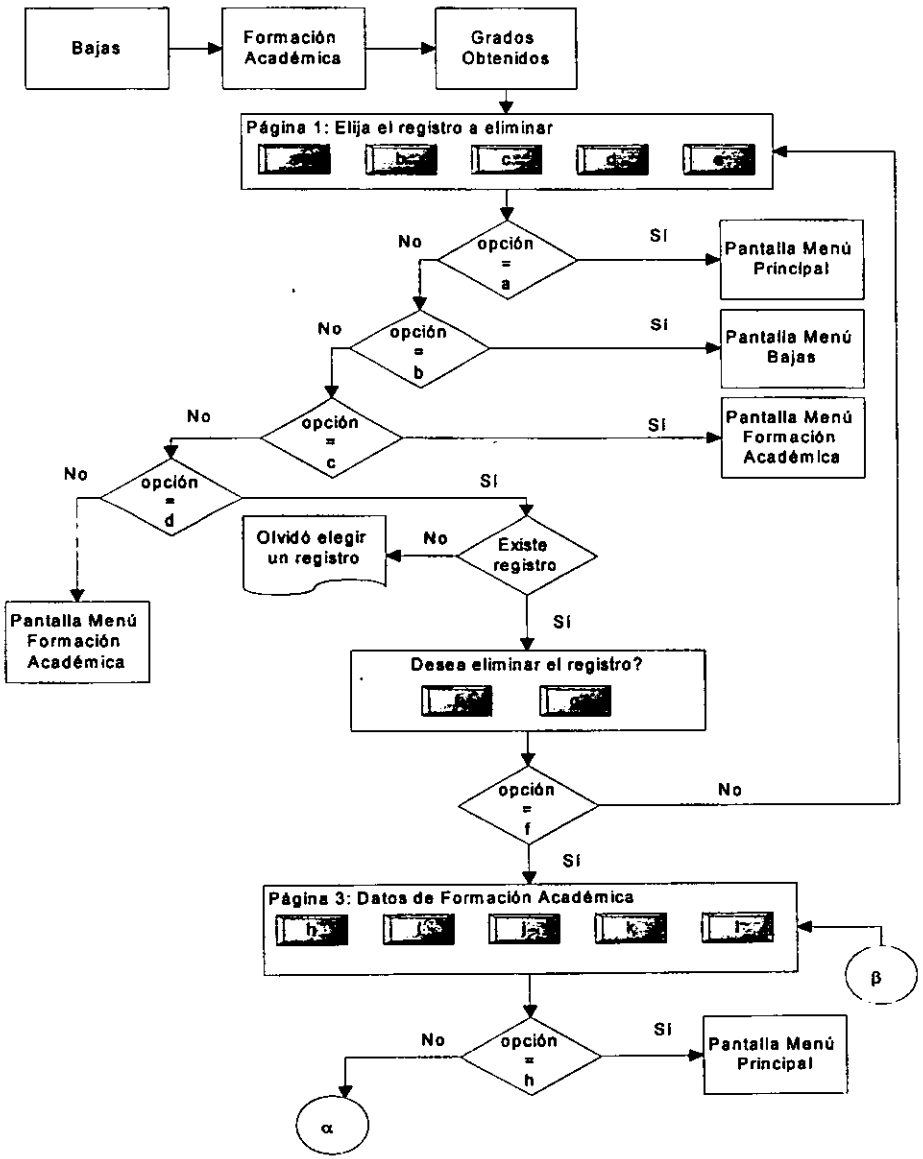
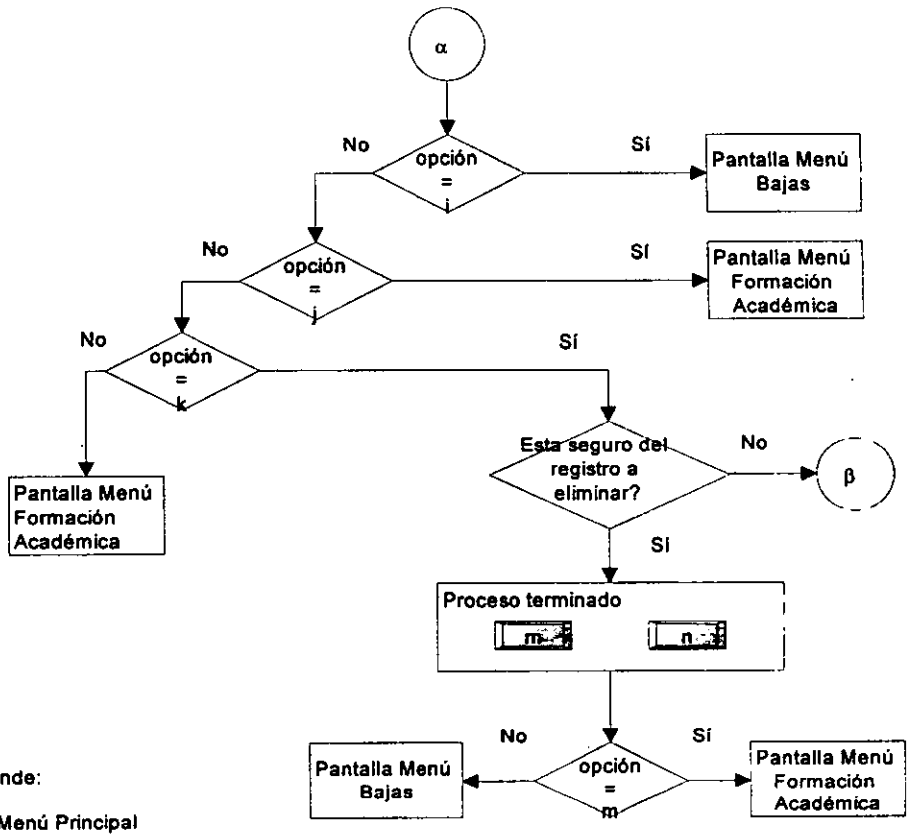


Figura 6: Diagrama de flujo del módulo altas y opción formación académica.

En la parte de bajas (Figura 7), se selecciona el registro que se desea eliminar, en caso que exista más de uno, posteriormente se muestran los datos más sobresalientes del registro en particular y pregunta si se está seguro del registro a eliminar, en caso de estarlo muestra toda la información que contiene el registro y permite eliminarlo.



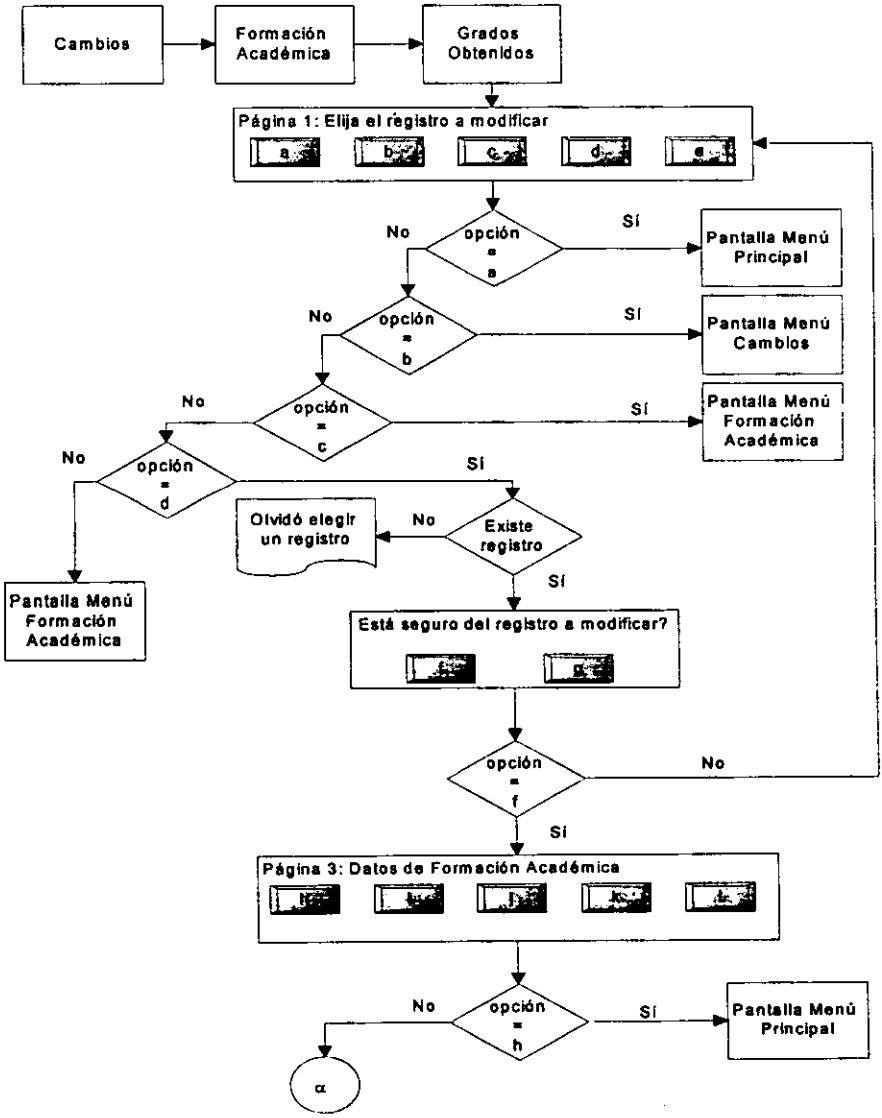


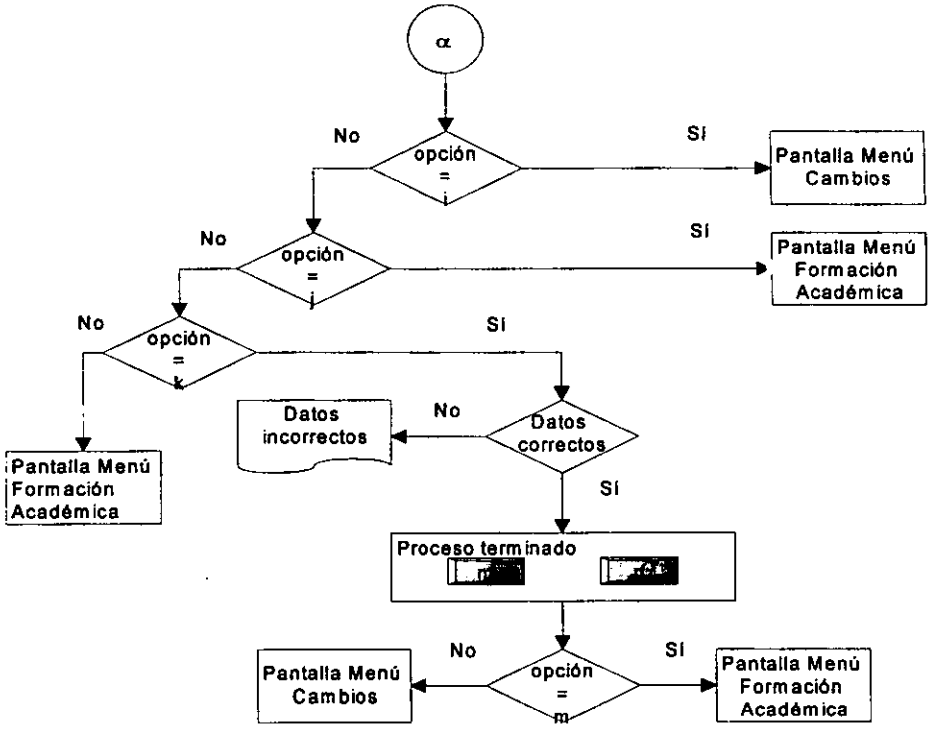
Donde:

a: Menú Principal
b: Bajas
c: Formación Académica
d: Aceptar
e: Salir
f: Aceptar
g: Regresar
h: Menú Principal
i: Bajas
j: Formación Académica
k: Eliminar
l: Salir
m:Regresar
n: Salir

Figura 7: Diagrama de flujo del módulo bajas y opción formación académica en el rubro de Grados Obtenidos

En la parte de cambios (Figura 8), se selecciona el registro que se desea modificar, en caso que exista más de uno, posteriormente se muestran los datos más sobresalientes del registro en particular que haya escogido y pregunta si se encuentra seguro del registro a modificar, en caso de estarlo muestra toda la información que contiene el registro y ahí se podrán seleccionar aquellos datos que se consideren prudentes para actualizarlo posteriormente.





Donde:

a: Menú Principal
b: Cambios
c: Formación Académica
d: Aceptar
e: Salir
f: Aceptar
g: Regresar
h: Menú Principal
i: Cambios
j: Formación Académica
k: Actualizar
l: Salir
m:Regresar
n: Salir

Figura 8: Diagrama de flujo del módulo cambios y opción formación académica en el rubro de Grados Obtenidos

En el capítulo siguiente se presenta de manera general el análisis, diseño y desarrollo del sistema BDAII Web.

CAPÍTULO 3

ANÁLISIS, DISEÑO Y DESARROLLO DEL SISTEMA DE INFORMACIÓN CURRICULAR UTILIZANDO HERRAMIENTAS WEB.

3.1 Análisis

El Instituto de Ingeniería dispone de una Base de Datos con la información curricular de su planta académica. Disponer de un sistema en WWW que permita la consulta y la alimentación de la información por parte de los académicos es altamente atractivo para el Instituto. Por lo anterior es necesario migrar las aplicaciones de consulta y alimentación de la Base de Datos Académica del Instituto de Ingeniería, desarrollando un sistema que permita a todos los académicos consultar de manera rápida y flexible su curriculum vitae, así como realizar la actualización del mismo. Dicha información puede ser consultada por el personal académico del Instituto de Ingeniería que tenga acceso a Internet.

3.1.1 Objetivo

Desarrollar una aplicación para consultar y alimentar a la Base de Datos Académica del Instituto de Ingeniería.

Para cumplir con el objetivo presentado anteriormente, la BDAII se compone de los siguientes módulos:

- Módulo de Altas: En este módulo, el usuario puede ingresar su información curricular a través de los diversos apartados de los cuales se compone este módulo, es importante que el usuario proporcione correctamente sus datos ya que a excepción del apartado de Datos personales, el cual se encargará de capturar los datos el administrador de la BDAII, la demás información será responsabilidad de cada investigador.

- Módulo de Bajas: En este módulo, el sistema permitirá al usuario eliminar el o los registros que así desee de cada diferente apartado. Es importante destacar que una vez que elimine el registro ya no será el sistema de recuperar la información a menos que vaya al módulo de altas y desde ahí ingrese de nuevo la información.

- Módulo de Cambios: El usuario de la BDAII tendrá la oportunidad en este módulo de actualizar alguna información de algún registro en específico, sin necesidad de ingresar nuevamente la información.

- Módulo de Consultas: A través de este módulo el usuario podrá visualizar toda su información de la BDAII que ha dado de alta o en su defecto que haya modificado, seleccionando el registro de acuerdo al apartado en donde se encuentre.

- Opción de Semblanza: En este módulo el usuario será capaz de consultar una breve descripción donde proporcionó su línea de investigación de los últimos años, además

de todos aquellos aspectos que considera sobresalientes y que no se contemplaron en ninguno de los apartados anteriores.

- Opción de Curriculum General: Este módulo es importante ya que aquí el investigador puede visualizar todo su curriculum completo, en un formato preestablecido y de aquí permite guardar su información en disco o simplemente mandarlo a imprimir.

- Módulo de Ayuda: En este módulo se cuenta con la ayuda que permitirá al usuario situarse en alguna página en específico de la cual desconozca su forma de llenado, este módulo funciona para todas y cada una de las páginas de captura de la BDAII.

- Opción de Comentarios: En este módulo se permiten los comentarios y sugerencias para la realimentación del sistema y su adecuado funcionamiento.

- Opción de Salir: Este módulo simplemente permite salir del sistema.

3.2 Diseño de las páginas

El sistema propuesto está basado en una arquitectura Cliente/Servidor[2], en el cual el cliente es el sistema Web para consulta y alimentación de la BDAII y el servidor se constituye de un servidor Web y un servidor de base de datos relacionales Sybase.

A continuación presentamos un esquema general del funcionamiento de cada uno de los módulos que componen al sistema BDAII.

3.2.1 Web

En las Figuras: 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 y 3.8 se muestran los esquemas elaborados para la interfaz del Web y la simbología utilizada, para cada uno de los módulos y submódulos que conforman el sistema BDAII.

[2] Para mayor referencia consultar el apéndice en el inciso A.

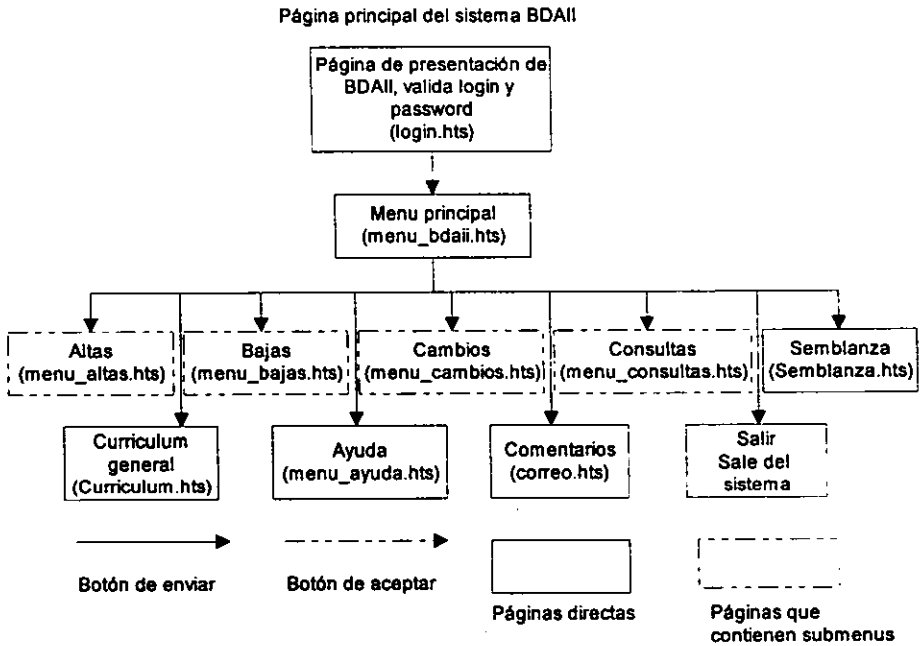


Fig. 3.1 Esquema que muestra como está constituida la página principal del sistema BDAII

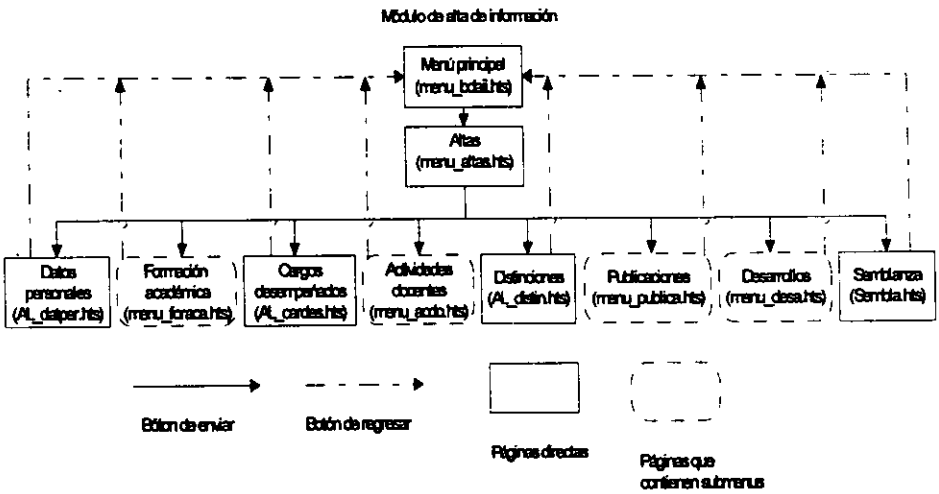


Fig 3.2 Esquema del módulo de altas

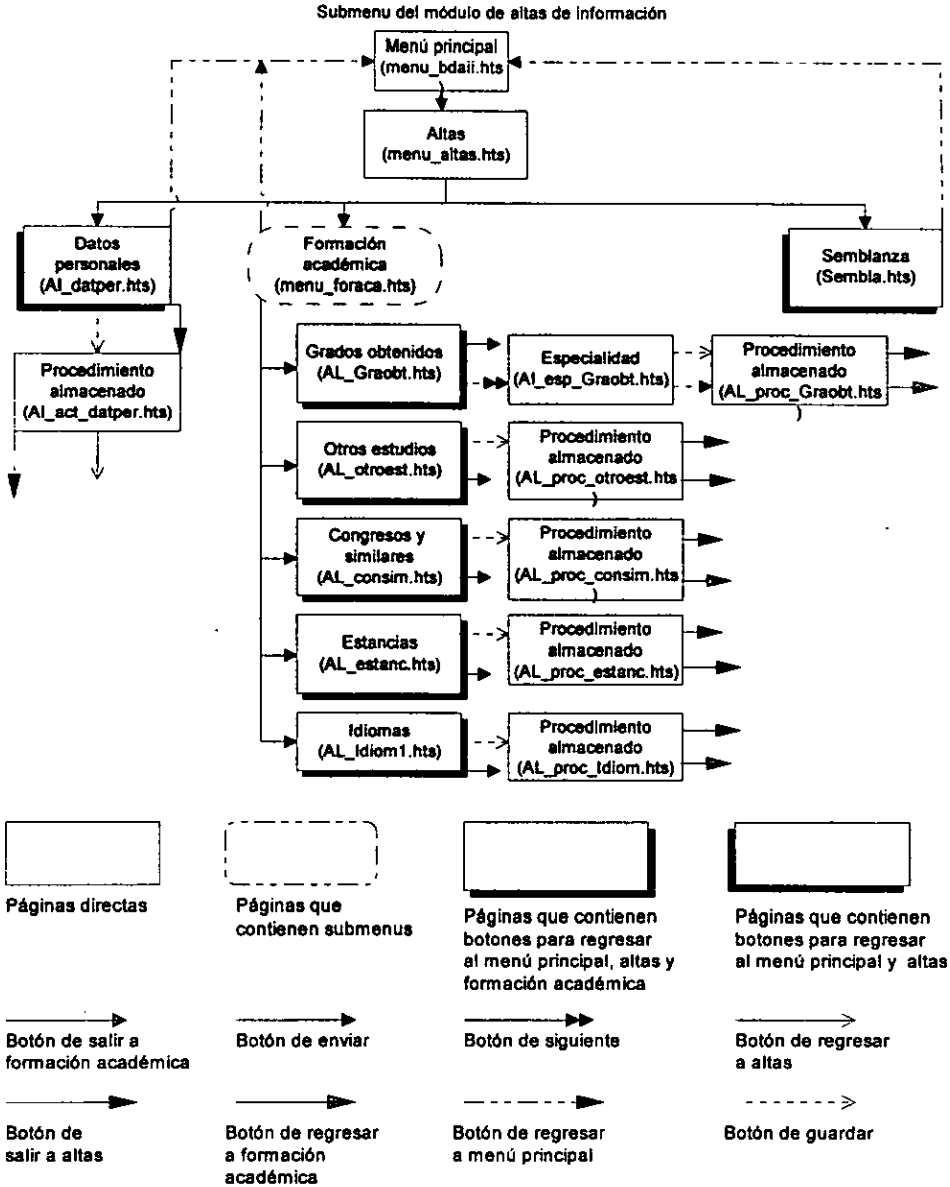


Fig. 3.3. Esquema que muestra el submódulo de formación académica para el módulo de altas

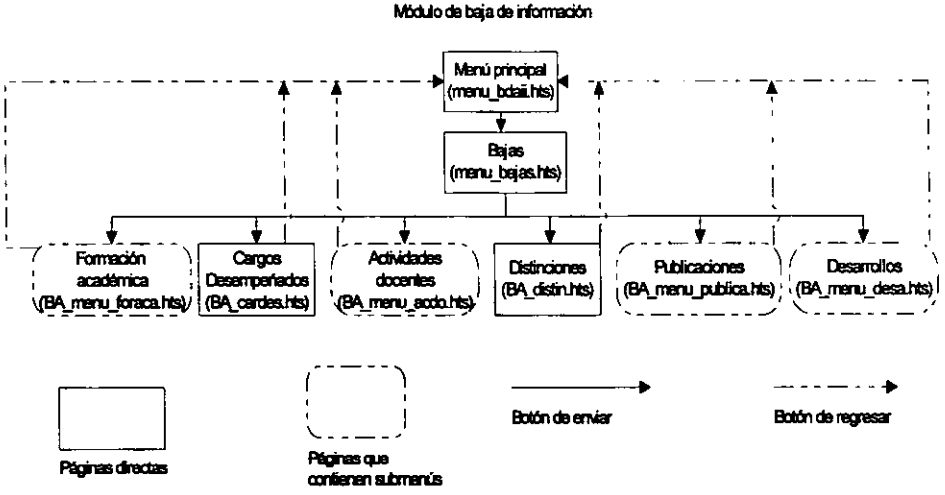


Fig. 3.4 Esquema del módulo de bajas

Como se observa en la Figura 3.4, el módulo de bajas no contempla ni los Datos Personales ni el rubro de Semblanza. Los primeros son responsabilidad del administrador de la Base de Datos, solo él podrá crear un usuario y, en su caso, darlo de baja. Obviamente la modificación de información si es posible. En el caso de la semblanza, solo es posible modificar su contenido.

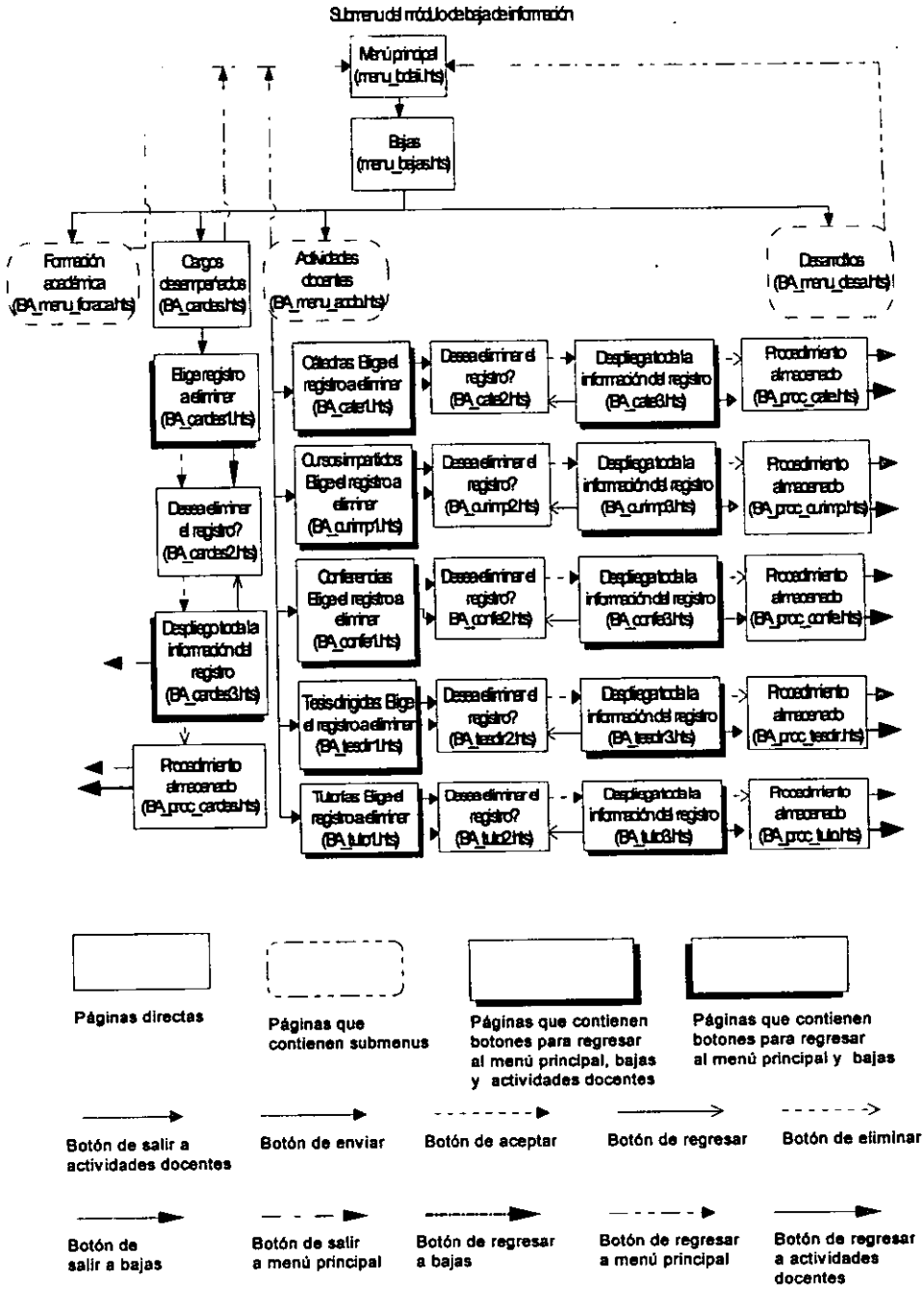


Fig. 3.5. Esquema que muestra el submódulo de actividades docentes para el módulo de bajas

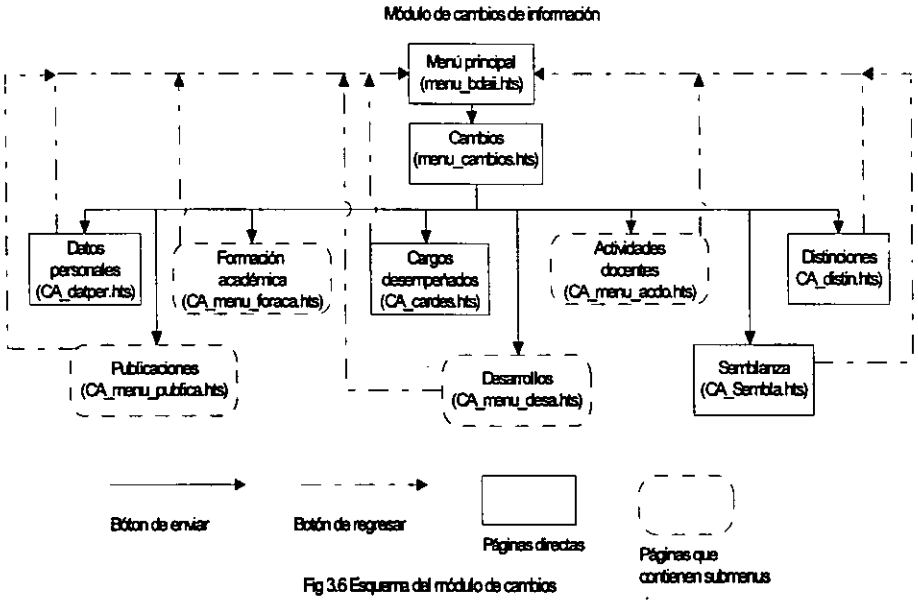


Fig 3.6 Esquema del módulo de cambios

Como se observa en la Figura 3.6, en el módulo de cambios todo podrá ser modificado, incluso hasta los datos personales.

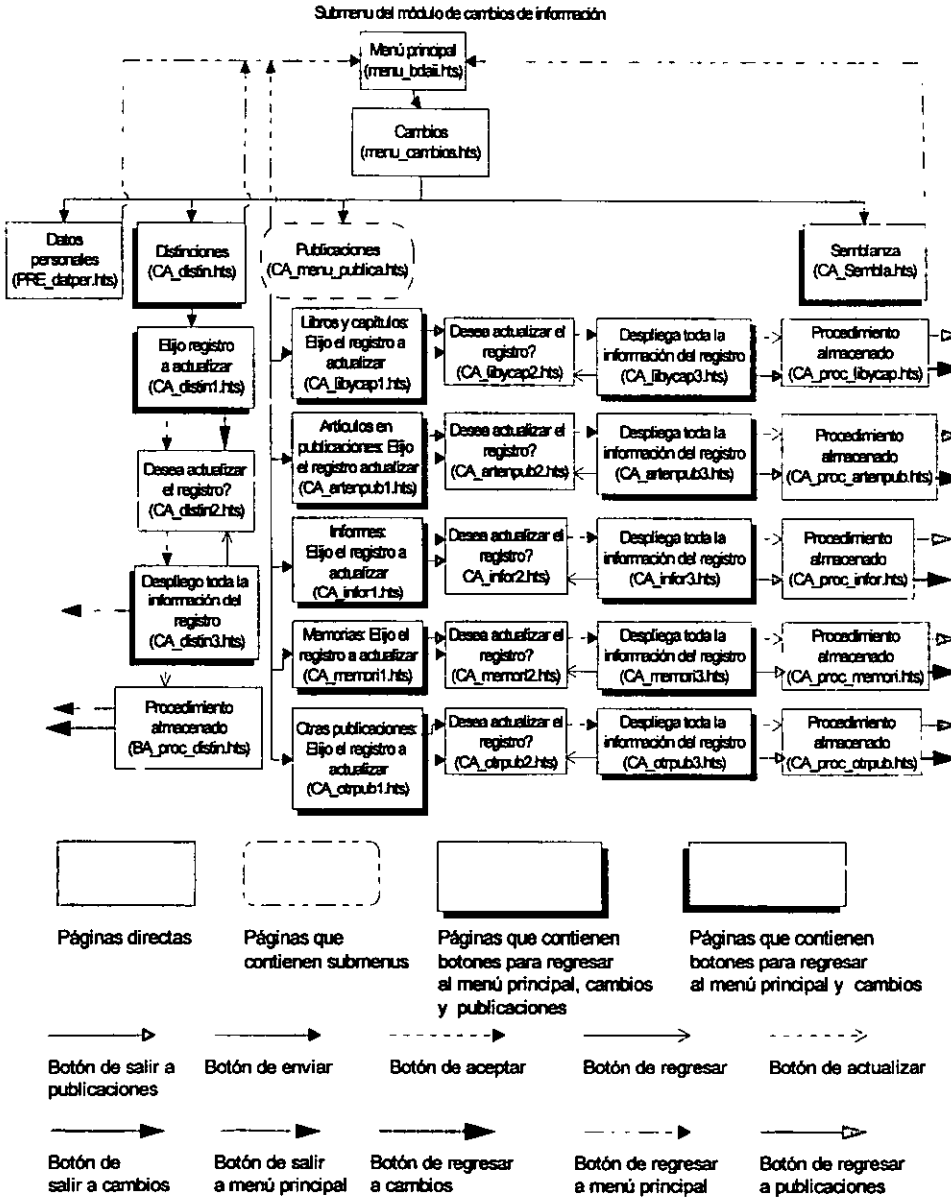


Fig. 3.7 Esquema que muestra el submódulo de publicaciones para el módulo de cambios

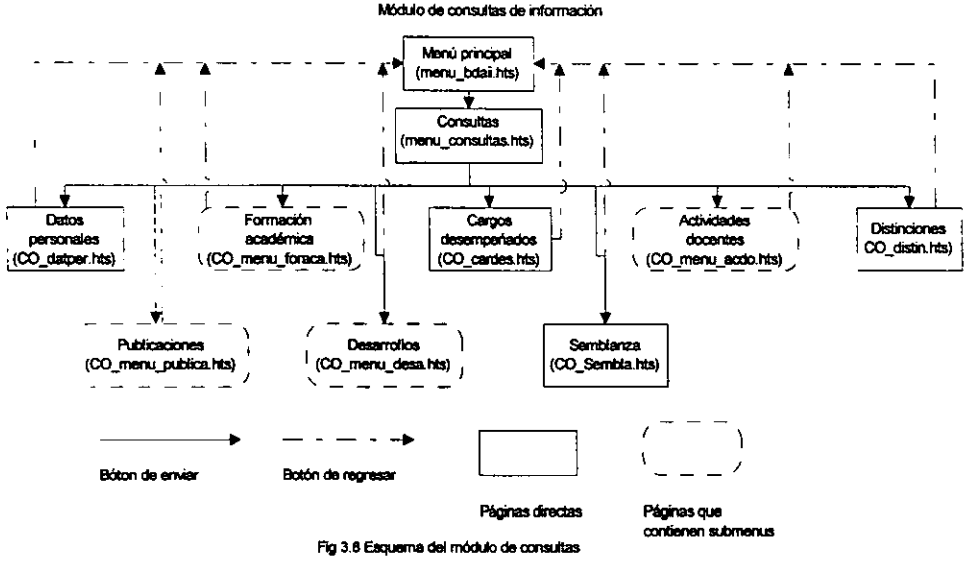


Fig 3.8 Esquema del módulo de consultas

Para desarrollar la interfaz del sistema y de la base de datos, se necesitó ayuda de herramientas que facilitarán el trabajo. Se aprovecharon las herramientas con las que se cuenta en el Instituto de Ingeniería.

Para el desarrollo de las páginas Web, se utilizó un Servidor de Web Apache[3], el cual permite atender las solicitudes de las páginas que se encuentran en el home page y con Netscape Comunicator como browser, el cual proporciona un editor de texto para crear páginas HTML[4].

Además se utilizó Web.sql[5], el cual es una buena opción en el desempeño de las bases de datos ya que las liga directamente con el servidor, mejorando la integración entre el servidor Web y la base de datos, evitando problemas de conexión, proporcionando rápidos accesos a la base de datos relacional y con ello un mejor tiempo de respuesta desde el World Wide Web.

[3] Para mayor información consultar el apéndice en el inciso F.
[4] Para mayor información consultar el apéndice en el inciso B.
[5] Para mayor información consultar el apéndice en el inciso I.

El siguiente paso en el desarrollo fue la programación de la página del Web. Este proceso consiste en traducir el diseño lógico de un sistema, a código fuente; para ello, se utilizaron herramientas de software como Web.sql, Perl, HTML, JavaScript y Transact SQL. A continuación se mostrará el desarrollo de las páginas más representativas del funcionamiento del sistema, para lograrlo se procedió de la siguiente manera:

La Figura 3.10 muestra la página de presentación.

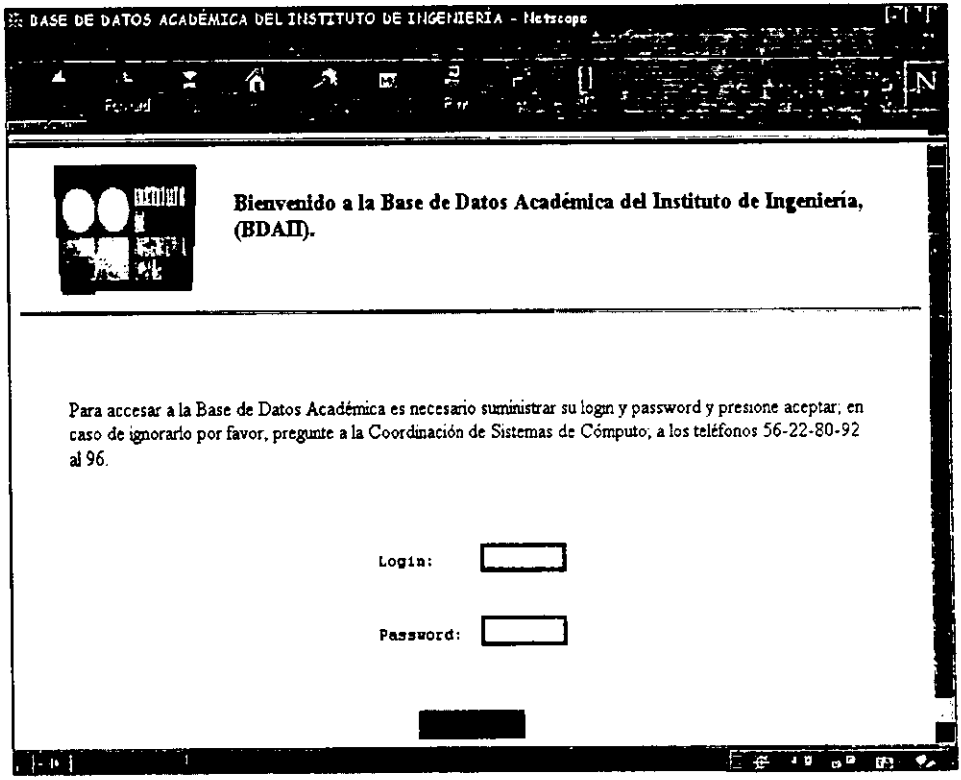


Figura 3.10 Página que representa la entrada al sistema de BDAII.

Para la realización de la página se utilizó código HTML, JavaScript, Perl y Web.sql. Con HTML se creó el cuerpo del documento: logotipo del Instituto de Ingeniería, título de la página, color de fondo, tamaño de la letra y una breve descripción del funcionamiento del sistema, así como la función que realiza dicha página.

Dentro de la forma HTML se insertó código JavaScript a fin de capturar el login y password. Este código verifica que ni el login ni el password estén vacios. La solicitud del login y password evita accesos no autorizados. A continuación se muestra dicho código:

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute; MICA DEL INSTITUTO DE INGENIER&Iacute;
A</TITLE>
</HEAD>
 <BODY  BGCOLOR="white" TEXT="#000000"
LINK="#rrggbb" VLINK="#rrggbb" ALINK="#rrggbb">
<P><P>

<TABLE><TD><A HREF="logo.gif"></A>
<IMG SRC="logo.gif" WIDTH=150 HEIGHT=100 BORDER=0 HSPACE=10><BR></TD>

<TD><H3>Bienvenido a la Base de Datos Acad&eacute;mica del Instituto de
Ingenier&iacute;a, (BDAII). </H3></TD></TABLE>
<HR><BR>

<script languaje="JavaScript">
function mensaje(u_name, passw){
     if ((u_name == "") || (passw == "")) {
     alert("Favor de teclear sus datos");
     document.forma.u_name.focus();
}
else{
if(confirm("Tecleo correctamente su login y password?")){
document.forma.submit();
}
}
}

function valida() {
if (confirm("Son correctos tus datos?")){
document.forma.submit();
}
}
</script>

</HEAD>
<BODY BGCOLOR="WHITE" TEXT="#000000" LINK="#rrggbb" VLINK="#rrggbb"
ALINK="#rrggbb">
<FORM METHOD="POST" ACTION="menu_bdaii.hts" NAME="forma"><BR><BR><P>
<BLOCKQUOTE>Para accesar a la Base de Datos Acad&eacute;mica del
Instituto de Ingenier&iacute;a es necesario suministrar su login y
password y presione aceptar; en caso de ignorarlo por favor, pregunte a
la Coordinación de Sistemas de C&oacute;mputo; a los teléfonos 56-22-80-
92 al 96. </BLOCKQUOTE><BR><BR><P>
<CENTER>
<PRE>Login:      <INPUT TYPE="TEXT" NAME="u_name" SIZE="8"
MAXLENGTH=15></PRE><BR><P>
```

```
<PRE>Password:  <INPUT TYPE="password" NAME="passw" SIZE="8"
MAXLENGTH=15></PRE><BR><P>
<INPUT TYPE="button" VALUE="ACEPTAR" NAME="aceptar"
onClick="mensaje(document.forma.u_name.value,
document.forma.passw.value)">
</CENTER>
</FORM>
</BODY>
</HTML>
```

La Figura 3.11 muestra la página de menú principal de una página Web.

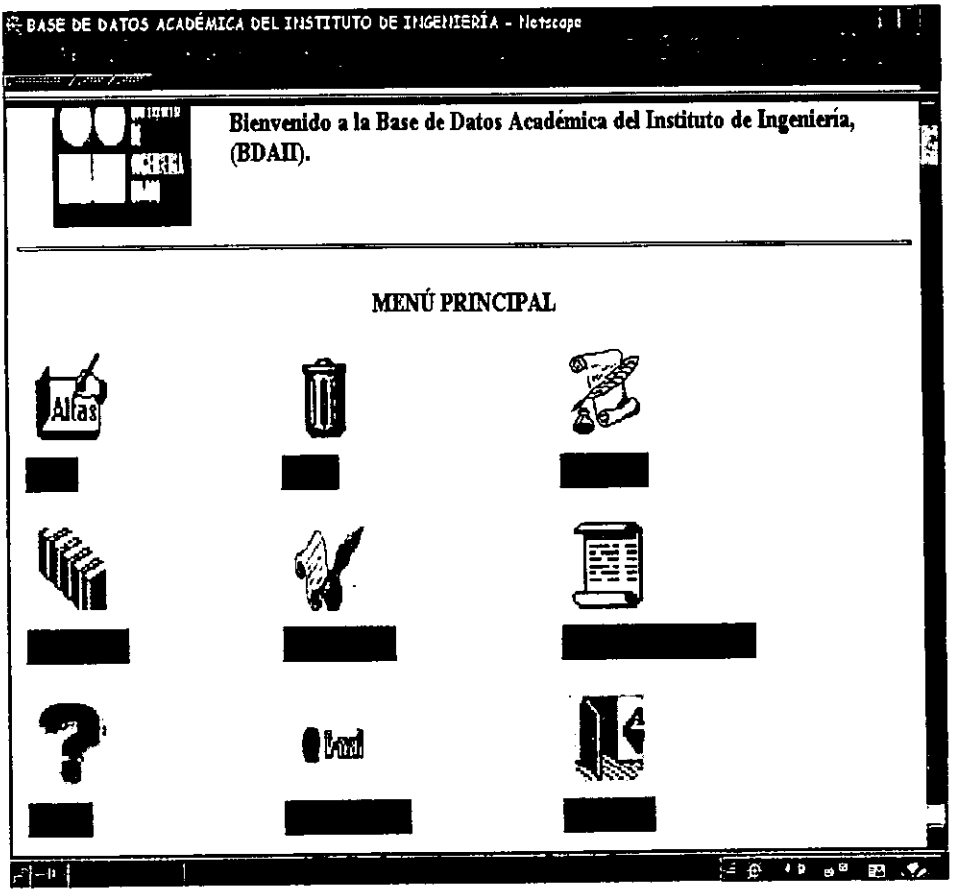


Figura 3.11 Página del menú principal del sistema BDAII

A continuación se presenta el código que conforma la página del menú principal del sistema BDAII.

Aquí básicamente se realiza la conexión a la Base de Datos para obtener el error del cliente y del servidor, en caso de que ocurra.

El resto de la página crea el menú principal de la BDAII, brindando la posibilidad de seleccionar alguna opción que permite trasladarse a una página diferente. A su vez se obtiene la clave del investigador, lo cual asegura que la información que se está accesando sea la que corresponde al usuario.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE>
</HEAD>
<BODY BGCOLOR="white" TEXT="black" LINK="white" VLINK="#23238E">
<P><P>
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>

<SYB TYPE=PERL>
ct_callback(CS_CLIENTMSG_CB, 'client_err');
ct_callback(CS_SERVERMSG_CB, 'server_err');

$SRV="SYBASE11";
$ws_db=ct_connect($u_name, $passw, $SRV);

sub client_err{
   local($layer, $origin, $severity, $errno, $msg, $omsg)=@_;
   print "<P><HR><STRONG>CLIENT ERROR OCURRED: </STRONG>\n";
   print "<P>Client error info: <BR>\n";
   printf "Error: Layer=%ld Origin=%ld Severity=%ld Number=%ld <BR>\n",
   $layer, $origin, $severity, $errno;
   printf "Message text: %s <BR>\n", $msg;

   if (defined($osmsg)){
       printf "OS Message '%s' <BR>\n", $osmsg;
   }
   print "<P><HR>";
   CS_SUCCEED;
}

sub server_err{
   local($cmd, $errno, $severity, $state, $line, $server, $proc,
$msg)=@_;
   print "<P><HR><STRONG>SERVER ERROR OCURRED:</STRONG>\n";
   print "<P>Server error info: <BR>\n";
   printf "Cmd=%s <BR>\n", $cmd;
   printf "Error: Number=%ld Severity=%ld State=%ld Line=%ld <BR>\n",
   $errno, $severity, $state, $line;
   if (defined($server)){
   printf "Server '%s' <BR>\n", $server;
```

```
}

if (defined($proc)){
   printf "Stored procedure '%s' <BR>\n", $proc;
   }
   printf "Message text: %s <BR>\n", $msg;
   print"<P><HR>";
   CS_SUCCEED;
}

if($layer=4 && $origin=1 && $severity=4 && $errno=44 && $errno=4002   &&
$severity=14 && $state=1 && $line=1){
print "Error: Falló su login y password ";
}
if($layer=4 && $origin=1 && $severity=4 && $errno=44){
print "Error: Fallo su login y password desde el cliente";
}
if($errno=4002  && $severity=14 && $state=1 && $line=1){
print "Error: Fallo su login y password desde el servidor";
}
if ($ws_db eq "") {
print <<ETI;
</FONT>
<FONT COLOR="FF0000">
<H3><CENTER>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</CENTER></H3>
<CENTER><INPUT TYPE="button" VALUE="REGRESAR" NAME="salir"
onClick="history.back()"></CENTER>
</FONT>
ETI
print $ws_db;

ws_error ("Fall&oacute; el login.");
     }
          else{
print <<ETI;
</FONT>
ETI

ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\")", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");

ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;

<TABLE><TD><A
HREF="/usr/apache/apache_1.2.5/htdocs/websql.dir/proyecto/logo.gif"></A>
<IMG SRC="logo.gif" WIDTH=150 HEIGHT=100 BORDER=0 HSPACE=10><BR></TD>
<TD><H3>Bienvenido a la Base de Datos Acad&eacute; mica del Instituto de
Ingenier&iacute;a, (BDAII). </H3></TD></TABLE>
<HR><BR>
```

```
<CENTER><H3>MEN&Uacute PRINCIPAL</H3></CENTER>
<TABLE ALIGN="LEFT">
<TR>
<TH ALIGN="LEFT"><IMG SRC="altas.gif" ALT="Altas" HEIGHT=60 WIDTH=60
HSPACE=10>
</TH>

<TH ALIGN="LEFT"><IMG SRC="basurero.gif" ALT="Bajas" HEIGHT=60 WIDTH=60
HSPACE=10></TH>
<TH ALIGN="LEFT"><IMG SRC="scroll2.gif" ALT="Cambios" HEIGHT=60 WIDTH=60
HSPACE=10>
</TH><TR>
<TD WIDTH=200 ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" action="menu_altas.hts">
<INPUT TYPE="SUBMIT" VALUE="Altas" NAME="altas">
ETI
ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</FORM>

<TD ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" action="menu_bajas.hts">
<INPUT TYPE="SUBMIT" VALUE="Bajas" NAME="Bajas">
ETI
ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</FORM>
<TD ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" ACTION="CA_menu_cambios.hts">
<INPUT TYPE="SUBMIT" VALUE="Cambios" NAME="Cambios">
ETI
ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</FORM>
</TD>
</TR>
<TR>
<TH ALIGN="LEFT"><IMG SRC="books01.gif" ALT="Consultas" HEIGHT=60
WIDTH=60 HSPACE=10>
</TH>
<TH ALIGN="LEFT"><IMG SRC="note.gif" ALT="SEMBLANZA" HEIGHT=60 WIDTH=60
HSPACE=10></TH>
<TH ALIGN="LEFT"><IMG SRC="scroll1.gif" ALT="CURRICULUM  GENERAL"
HEIGHT=60 WIDTH=60 HSPACE=10>
</TH><TR>
<TD WIDTH=100 ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" action="menu_consultas.hts">
<INPUT TYPE="SUBMIT" VALUE="Consultas" NAME="Consultas">
ETI
```

```
ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</FORM>

<TD ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" ACTION="Semblanza.hts">
<INPUT TYPE="SUBMIT" VALUE="Semblanza" name="semblanza">
ETI
ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</FORM>
<TD ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" ACTION="Curriculum.hts">
<INPUT TYPE="SUBMIT" VALUE="Curriculum General" name="Curri_gral">
ETI
ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</FORM>
</TD>
</TR>

<TR>
<TH ALIGN="LEFT"><IMG SRC="ayuda.gif" ALT="AYUDA" HEIGHT=60 WIDTH=60
HSPACE=10></TH>
<TH ALIGN="LEFT"><IMG SRC="e-mail.gif" ALT="COMENTARIOS" HEIGHT=60
WIDTH=60 HSPACE=10></TH>
<TH ALIGN="LEFT"><IMG SRC="dooring.gif" ALT="SALIR" HEIGHT=60 WIDTH=60
HSPACE=10>
</TH>
<TR>
<TD WIDTH=400 ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" ACTION="menu_ayuda.hts">
<INPUT TYPE="SUBMIT" VALUE="Ayuda" NAME="Ayuda">
</FORM>
<TD WIDTH=400 ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST" ACTION="correo.hts">
<INPUT TYPE="SUBMIT" VALUE="Comentarios" NAME="comentarios">

</FORM>
<TD WIDTH=400 ALIGN="TOP" ALIGN="LEFT">
<FORM METHOD="POST ACTION="CO_Salir.hts">
<INPUT TYPE="button" VALUE="S A L I R" NAME="salir"
onClick="history.back(-1)">
</FORM>
</TD>
</TR>
</TABLE>
ETI
```

```
}
</SYB>
</FORM>
</BODY>
</HTML>
```

La Figura 3.12 muestra un ejemplo de la página realizada para el módulo de altas.

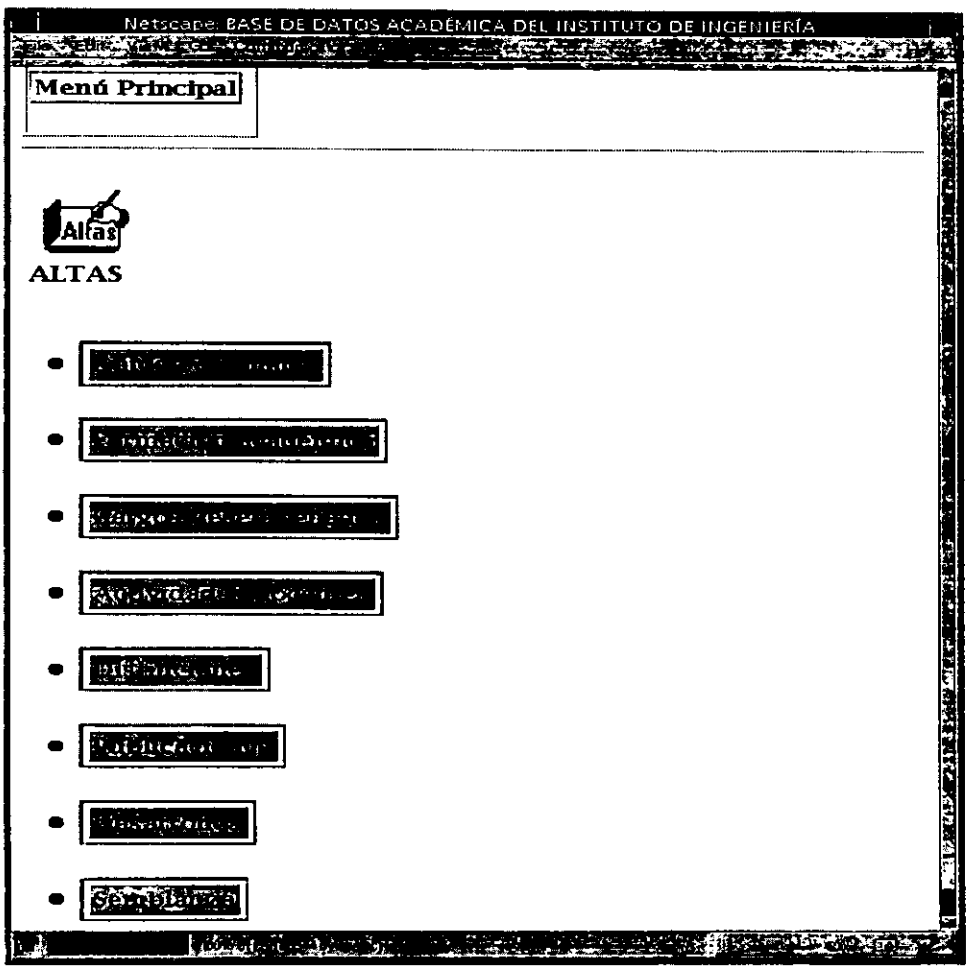


Figura 3.12 Página del menú de altas del sistema BDAII

Por medio de código HTML se activó cada uno de los botones que permiten trasladarse a través de las diferentes opciones para trabajar en el menú de altas. Fue necesario obtener datos del investigador, los cuales se extrajeron de la base de datos.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE></HEAD>
<BODY BGCOLOR="white" TEXT="#000000"
LINK="#rrggbb" VLINK="#rrggbb" ALINK="#rrggbb">
<FORM METHOD="POST" action="AL_Fordatpe.hts">
<P><P>

<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>

<TABLE BORDER = "1">
<TR><TH><FORM METHOD="POST" ACTION="menu_bdaii.hts">
<INPUT TYPE="button" NAME="regresa" VALUE="Men&uacute; Principal"
onClick="history.back()">
</FORM>
</TH>
</TR>
</TABLE>
<HR>
<BR><P>

<TABLE>
<TR><TH><IMG SRC="altas.gif" ALT="Altas" HEIGHT=60 WIDTH=60
HSPACE=10></TH>
<TH></TH>
<TH></TH></TR>
<TD ALIGN="TOP" ALIGN="LEFT">
<H4>ALTAS</H4>
</TD></TABLE>
<UL>
<FORM METHOD="POST" action="AL_datper.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");

$sql_stmt = qq!select cl_p_naci from dat_person2 where cl_inves=
convert(smallint,\"$cl_inves\")!;
if (( $rc = ct_sql( $ws_db, $sql_stmt )) != CS_SUCCEED ) {
       ws_error ("Unable to process database request.");
}

while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED) {
   RESULTS: {
       if ($result_type == CS_CMD_DONE) {
```

```
last RESULTS;
       }

if ($result_type == CS_CMD_FAIL) {
       last RESULTS;
       }
       if ( $result_type == CS_ROW_RESULT ) {
        while (@row = ct_fetch ( $ws_db )) {
        $var=@row[0];
        }
       last RESULTS;
      }
           }
}
if ($ret == CS_FAIL) {
ct_cancel(CS_CANCEL_ALL);
ws_error("A database connection error occurred");
print "error\n";
}
</SYB>
<SYB TYPE=PERL>
ws_sql($ws_db, "select cl_pais from pais where cl_pais=\"$var\"", "<INPUT
TYPE=\"hidden\" NAME=\"pais\" VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<SYB TYPE=PERL>
$sql_stmt = qq!select cl_p_nacio from dat_person2 where
cl_inves=convert(smallint,\"$cl_inves\")!;
if (( $rc = ct_sql( $ws_db, $sql_stmt )) != CS_SUCCEED ) {
       ws_error ("Unable to process database request.");
}

while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED) {
   RESULTS: {
       if ($result_type == CS_CMD_DONE) {
      last RESULTS;
       }
      if ($result_type == CS_CMD_FAIL) {
       last RESULTS;
       }
       if ( $result_type == CS_ROW_RESULT ) {
        while (@row = ct_fetch ( $ws_db )) {
        $var1=@row[0];
        }
       last RESULTS;
      }
           }
}
if ($ret == CS_FAIL) {
ct_cancel(CS_CANCEL_ALL);
ws_error("A database connection error occurred");
print "error\n";
}
ws_sql($ws_db, "select clave from nacional where clave=\"$var1\"",
"<INPUT TYPE=\"hidden\" NAME=\"nacionalidad\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
```

```
</SYB>

<SYB TYPE=PERL>
$sql_stmt = qq!select cl_pais from dat_person2 where cl_inves=
convert(smallint,\"$cl_inves\")!;
if (( $rc = ct_sql( $ws_db, $sql_stmt )) != CS_SUCCEED ) {
       ws_error ("Unable to process database request.");
}

while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED) {
   RESULTS: {
       if ($result_type == CS_CMD_DONE) {
      last RESULTS;
       }
      if ($result_type == CS_CMD_FAIL) {
       last RESULTS;
       }
       if ( $result_type == CS_ROW_RESULT ) {
        while (@row = ct_fetch ( $ws_db )) {
        $var=@row[0];
        }
       last RESULTS;
      }
           }
}
if ($ret == CS_FAIL) {
ct_cancel(CS_CANCEL_ALL);
ws_error("A database connection error occurred");
print "error\n";
}
</SYB>
<SYB TYPE=PERL>
ws_sql($ws_db, "select cl_pais from pais where cl_pais=\"$var\"", "<INPUT
TYPE=\"hidden\" NAME=\"pais_est\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
</SYB>
<LI><INPUT TYPE="SUBMIT" VALUE="Datos personales" NAME="dat_per"></LI>
</FORM>
<FORM METHOD="POST" action="menu_foraca.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Formaci&oacute;n acad&eacute;mica"
NAME="forma_acade"></LI>
</FORM>
<FORM METHOD="POST" action="AL_cardes.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<LI>
```

```
<INPUT TYPE="SUBMIT" VALUE="Cargos desempe&ntilde;ados"
NAME="cargos_desem"></LI>
</FORM>
<FORM METHOD="POST" action="menu_acdo.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Actividades docentes" NAME="activ_doc"></LI>
</FORM>
<FORM METHOD="POST" action="AL_distin.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<syb type=perl>
ws_sql($ws_db, "select cl_pais from pais where cl_pais=\"276\" ", "<INPUT
TYPE=\"hidden\" NAME=\"pais_est\" VALUE=\"%s\" SIZE=\"5\"
MAXLENGTH=\"5\">");
</syb>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Distinciones" NAME="distinciones"></LI>
</FORM>
<FORM METHOD="POST" action="menu_publica.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Publicaciones" NAME="publica"></LI>
</FORM>
<FORM METHOD="POST" action="menu_desa.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Desarrollos" NAME="desarrollos"></LI>
</FORM>
<FORM METHOD="POST" action="Sembla.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Semblanza" NAME="Semblanza"></LI>
</FORM>
```

```
</UL>
</BODY>
</HTML>
```

Está página en específico, es el mismo modelo tanto como para cambios como para consultas. En bajas la diferencia es que en el menú no aparece la opción de datos personales y semblanza, debido a que el usuario no puede eliminar estos datos del sistema directamente, para ello se cuenta con un campo que se llama fecha de baja en el cual se coloca su fecha de término en el Instituto de Ingeniería y de está manera ya no se encuentre entre los usuarios activos.

La Figura 3.13 muestra un ejemplo de las páginas realizadas para el módulo de altas, dentro del submódulo de publicaciones.

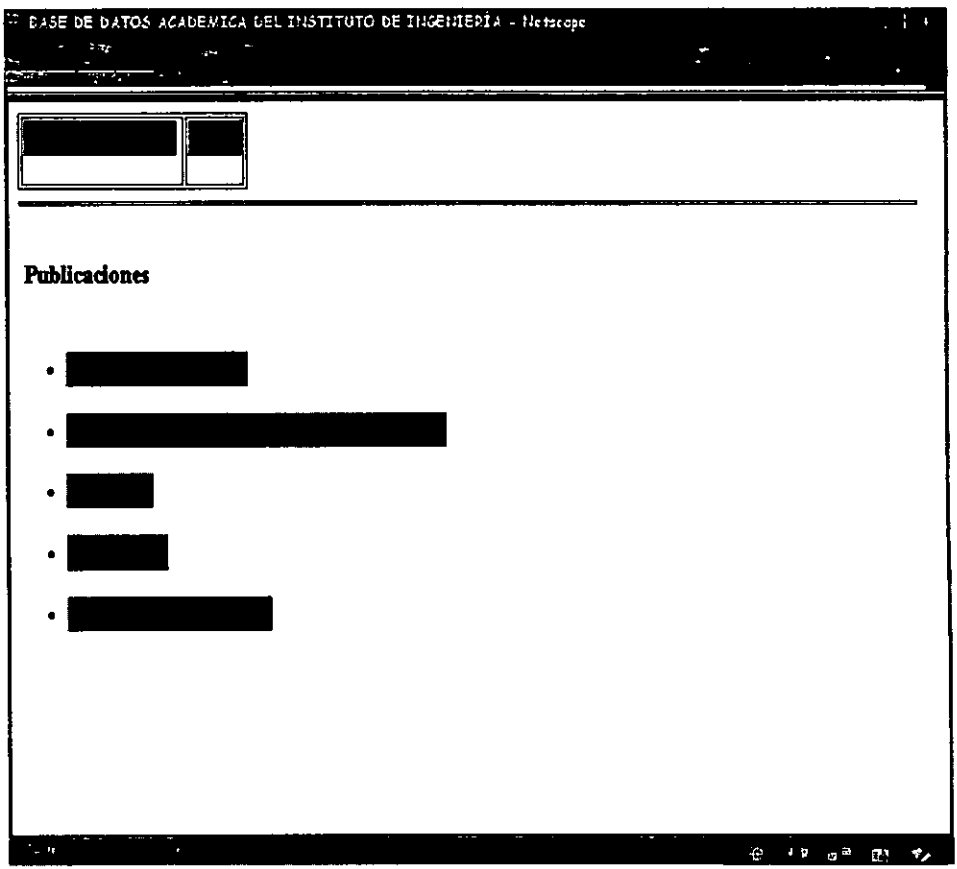


Figura 3.13 Página del menú altas dentro del submódulo publicaciones

En la parte superior izquierda de la pantalla se cuenta con dos botones los cuales permiten regresar directamente al menú de altas o al menú principal. Se utilizaron botones de JavaScript para regresar a las diferentes opciones antes mencionadas.

Dentro de la misma página se cuenta con opciones que permiten navegar hacia las diferentes páginas que conforman el menú de publicación. El código no se presenta ya que es similar al de la Figura 3.12.

Este mismo código se utilizó para los módulos de bajas, cambios y consultas, con la particularidad de que cada una de ellas se traslada a diferentes páginas.

La Figura 3.14 muestra una página Web del módulo de altas para el submódulo de publicaciones.

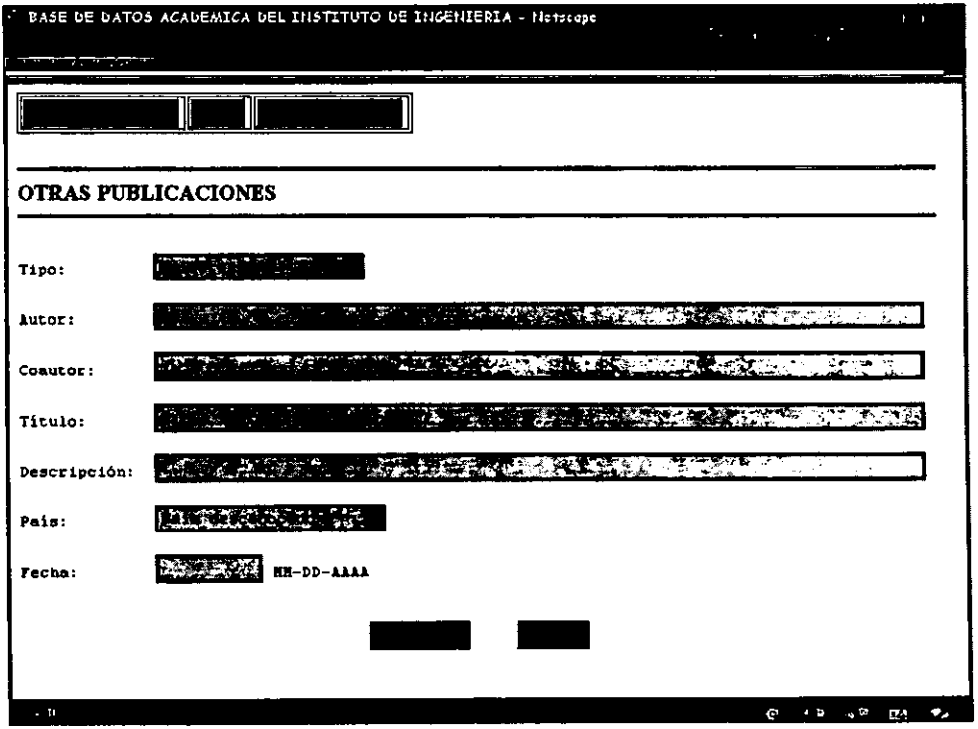


Figura 3.14 muestra la página de otras publicaciones para el submódulo de publicaciones dentro del módulo de altas.

El código utilizado se explica a continuación: por medio de una función en JavaScript, se pueden desplegar opciones que tienen que aparecer por default para el usuario, como la de País desconocido por ejemplo, lo cual se logra en el momento en que se carga la página. A su vez se encuentran otras funciones que permiten validar las cajas de texto. Cuando los campos han sido llenados por el usuario, se envía un mensaje para preguntar si sus datos son correctos, este mensaje se activa en el momento en que se hace un click en el botón de guardar, ya que de ahí se pasa a un procedimiento almacenado que guarda automáticamente los datos en el servidor de bases de datos Sybase.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE></HEAD>
<script languaje="JavaScript">
function cl_pais(pais_est){
i=0;
  for(;;)
  {

if (pais_est == document.frm_cur_imp.elements[9].options[i].value) {
         document.frm_cur_imp.elements[9].options[i].selected=true;
         break;
     }
      i++;
  }
}

function mensaje(nombre){
   if(nombre == ""){
       alert("Favor de teclear sus datos.\nNo puede dejar en blanco el
siguiente campo:\nTitulo");
       document.frm_cur_imp.nombre.focus();
   }
   else{
        if(confirm("Son correctos sus datos?")){
        document.frm_cur_imp.submit();
        }
   }
}
function valida() {
   if (confirm("Son correctos sus datos?")){
      document.frm_cur_imp.submit();
   }
}
</script>
<BODY onLoad="cl_pais($pais_est);" BGCOLOR=#FFFFFF><P>
<FORM METHOD="POST" ACTION="AL_ACT_AL_Publotpu.hts" name="frm_cur_imp">

<TABLE BORDER = "1">
<TR><TH>
<INPUT TYPE="button" VALUE="Men&uacute; Principal" NAME="menu_bdaii"
onClick="history.go(-3)">
</TH>
```

```
<TH><INPUT TYPE="button" VALUE="Altas" NAME="menu_altas"
onClick="history.go(-2)">
</TH>
<TH><INPUT TYPE="button" VALUE="Publicaciones" NAME="menu_publicacion"
onClick="history.back()">
</TH>
</TR>
</TABLE><P><P>

<H3><HR>OTRAS PUBLICACIONES<HR></H3>
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<PRE>
Tipo:          <SELECT NAME="tipo">
<SYB TYPE = PERL>
ws_sql($ws_db,"select tip_pub, des_pub from tipo_pub","<option
value=\"%s\">%s");
</SYB>
</SELECT><BR><P>
Autor:         <INPUT TYPE="text" NAME="autor" VALUE="" SIZE="79"
MAXLENGTH="100"><BR><P>
Coautor:       <INPUT TYPE="text" NAME="coautor" VALUE="" SIZE="79"
MAXLENGTH="255"><BR><P>
Título:        <INPUT TYPE="text" NAME="nombre" VALUE="" SIZE="79"
MAXLENGTH="100"><BR><P>
Descripción:  <INPUT TYPE="text" NAME="descripcion" VALUE="" SIZE="79"
MAXLENGTH="255"><BR><P>
País:          <SELECT NAME="pais">
<SYB TYPE = PERL>
ws_sql($ws_db,"select distinct pais.cl_pais, pais.d_pais from pais order
by pais.d_pais","<option value=\"%s\">%s");
</SYB>
</SELECT><BR><P>
Fecha:         <INPUT TYPE="text" NAME="fecha" VALUE="" SIZE="10"
MAXLENGTH="10"> MM-DD-AAAA<BR><P>

<CENTER><INPUT TYPE="button" VALUE="GUARDAR"
onClick="mensaje(document.frm_cur_imp.nombre.value)">      <INPUT
TYPE="BUTTON" VALUE="SALIR" NAME="SALIR"
onClick="history.back()"></CENTER>
</PRE>
</FORM>
</BODY>
</HTML>
```

La Figura 3.15 muestra una página que ejecuta con éxito un procedimiento almacenado.

Figura 3.15 muestra la página de un procedimiento almacenado

Esta parte del código es la parte más importante de toda la programación, ya que aquí es donde finalmente se guardan los valores de las variables que fueron tecleados en las cajas de texto y van a ser almacenadas en el servidor de bases de datos Sybase.

```
<HTML>
<HEAD>
<TITLE>ACTUALIZACION</TITLE>
</HEAD>
<BODY BGCOLOR=#FFFFFF><P>
<FORM>

<INPUT TYPE="hidden" NAME="cl_inves" VALUE="$cl_inves" SIZE="10"
MAXLENGTH=10>
<INPUT TYPE="hidden" NAME="tipo" VALUE="$tipo" SIZE="10" MAXLENGTH=10>
```

```
<INPUT TYPE="hidden" NAME="autor" VALUE="$autor" SIZE="100"
MAXLENGTH=100>
<INPUT TYPE="hidden" NAME="coautor" VALUE="$coautor" SIZE="255"
MAXLENGTH=255>
<INPUT TYPE="hidden" NAME="nombre" VALUE="$nombre" SIZE="100"
MAXLENGTH=100>
<INPUT TYPE="hidden" NAME="descripcion" VALUE="$descripcion" SIZE="255"
MAXLENGTH=255>
<INPUT TYPE="hidden" NAME="pais" VALUE="$pais" SIZE="10" MAXLENGTH=10>
<INPUT TYPE="hidden" NAME="fecha" VALUE="$fecha" SIZE="10" MAXLENGTH=10>

<SYB TYPE=PERL>
$fecha =~ s/(\/)+/\-/g;
$autor =~ s/\A(\s)+//;
$coautor =~ s/\A(\s)+//;
$nombre =~ s/\A(\s)+//;
$descripcion =~ s/\A(\s)+//;
$fecha =~ s/(\s)+//g;

if($fecha eq ""){
$fecha="01-01-2050";
}

if($nombre eq ""){
  print <<ETI;
  <CENTER><H4>Error en nombre, no se admiten campos vacios.</H4></CENTER>
ETI
}

if($fecha eq ""){
  print <<ETI;
  <CENTER><H4>Error en fecha, no se admiten campos vacios.</H4></CENTER>
ETI
}

@fecha=$fecha =~ /\d\d\d\d/g;
  for $token(@fecha){
      $F = $token;
  }
@mes=$fecha =~ /\A\d\d/g;
  for $token(@mes){
      $M = $token;
  }
      @dia=$fecha =~ /\-\d\d\-/g;
  for $token(@dia){
      $D = $token;
      $D=~tr/\-//d;
  }

$BISI="1";
if(($F % 4 eq "0" && $F % 100 ne "0")|| $F % 400 eq "0"){
      $BISI="0";
        if($M eq "02" && $D le "29"){
         }
         elsif($M eq "02" && $D gt "29"){
```

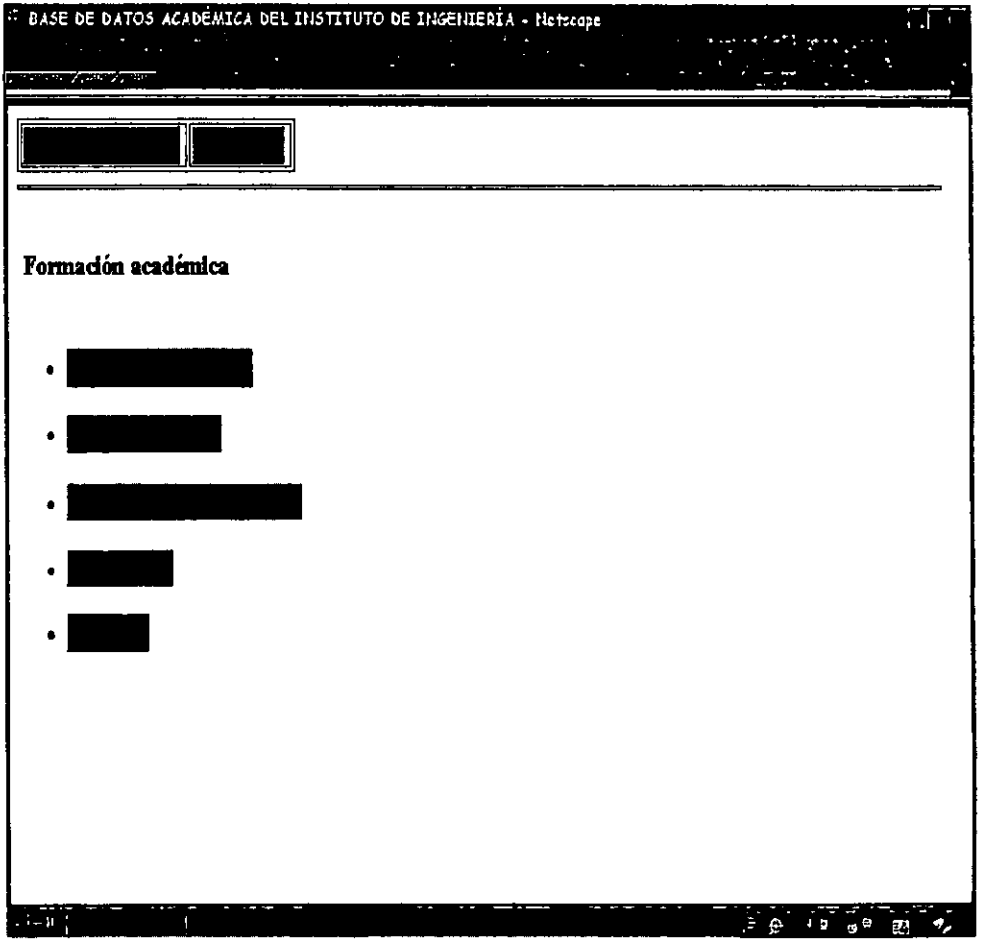
```
        print <<ETI;
  <CENTER><H4>Error en fecha, el mes de febrero no tiene mas de 29 días
en un año bisiesto.</H4></CENTER>
ETI
         }
}
elsif($BISI eq "1" && $M eq "02" && $D gt "28"){
  print <<ETI;
  <CENTER><H4>Error en fecha, el mes de febrero no tiene mas de 28 días,
en un año que no es bisiesto. </H4></CENTER>
ETI
}

if(($M eq "01" && $D gt "31")||($M eq "03" && $D gt "31")||($M eq "05" &&
$D gt "31")||($M eq "07" && $D gt "31")||($M eq "08" && $D gt "31")||($M
eq "10" && $D gt "31")||($M eq "12" && $D gt "31")){
print <<ETI;
  <CENTER><H4>Error en fecha, los meses de enero, marzo, mayo, julio,
agosto, octubre y diciembre no tiene un día posterior al 31.
</H4></CENTER>
ETI
}
if($M gt "12"){
print <<ETI;
  <CENTER><H4>Error en fecha, recuerde que no existe un mes posterior al
12. </H4></CENTER>
ETI
}
if($F lt "1900"){
print <<ETI;
<CENTER><H4>Error en fecha, recuerde que la fecha debe ser posterior a
1900. </H4></CENTER>
ETI
}
if($F gt "2060"){
print <<ETI;
  <CENTER><H4>Error en fecha, recuerde que la fecha debe ser anterior a
2060. </H4></CENTER>
ETI
}
if(($M eq "04" && $D gt "30")||($M eq "06" && $D gt "30")||($M eq "09" &&
$D gt "30")||($M eq "11" && $D gt "30")){
 print <<ETI;
  <CENTER><H4>Error en fecha, los meses de abril, junio, septiembre y
noviembre no tienen un día posterior a 30. </H4></CENTER>
ETI
  }

if($fecha !~ /\d\d(\-)\d\d(\-)\d\d\d\d/){
   print <<ETI;
   <CENTER><H4>No sé est&aacute; respetando el formato en fecha.
</H4></CENTER>
ETI
}
```

```
if(($nombre eq "")||($fecha eq "")||($BISI eq "0" && $M eq "02" && $D gt
"29")||($BISI eq "1" && $M eq "02" && $D gt "28")||($M eq "01" && $D gt
"31")||($M eq "03" && $D gt "31")||($M eq "05" && $D gt "31")||($M eq
"07" && $D gt "31")||($M eq "08" && $D gt "31")||($M eq "10" && $D gt
"31")||($M eq "12" && $D gt "31")||($M gt "12")||($F lt "1900")||($F gt
"2060")||($M eq "04" && $D gt "30")||($M eq "06" && $D gt "30")||($M eq
"09" && $D gt "30")||($M eq "11" && $D gt "30")||($fecha !~ /\d\d(\-
)\d\d(\-)\d\d\d\d/)){
  print <<ETI;
        <center><pre><blockquote><font color="ff0000"><H3>ERROR AL
INTRODUCIR DATOS!!!</H3></font></blockquote></pre>
<INPUT TYPE="BUTTON" NAME="regresa" VALUE="REGRESAR"
onClick="history.back()"></center>
ETI
}
else{
$res = ws_rpc($ws_db, "resp_bdaii.dbo.AL_otrpub",
{'@cl_inves' =>  {'value' => "$cl_inves", 'type' => CS_SMALLINT_TYPE },
          '@tip_pub'  => { 'value' => "$tipo", 'type' => CS_CHAR_TYPE },
          '@autor' =>  { 'value' => "$autor",  'type' => CS_CHAR_TYPE },
          '@coautor'  => { 'value' => "$coautor", 'type' => CS_CHAR_TYPE
},
          '@nombre' =>  { 'value' => "$nombre",  'type' => CS_CHAR_TYPE
},
          '@descripcion' => {'value' => "$descripcion", 'type' =>
CS_CHAR_TYPE },
          '@cl_pais'  => { 'value' => "$pais", 'type' => CS_CHAR_TYPE },
          '@fecha' => {'value' => "$fecha", 'type' => CS_DATETIME_TYPE
}});
if ($res == 0)
  {
       print <<ETI;
        <center><pre><blockquote><font color="ff0000">PROCESO
TERMINADO!!!</font><P>
<font color="ff0000">(</font>Su información ha sido guardada<font
color=\"ff0000\">)</font></blockquote></pre>
<INPUT TYPE="BUTTON" NAME="regresa" VALUE="REGRESAR"
onClick="history.go(-2)">
<input type="button" name="BACK" value="SALIR" onClick="history.go(-
3)"></center>
ETI
  }
else
   {
       print <<ETIQUETA;
        <center><pre><blockquote><font color="ff0000">ERROR!!!</font><P>
        <font color="ff0000"> (</font>¡Fallo en la alta!<font
color=\"ff0000\">)</font> </blockquote></pre>
<INPUT TYPE="BUTTON" NAME="regresa" VALUE="REGRESAR"
onClick="history.back()">
ETIQUETA
   }
}
</SYB>
</FORM>
```

```
</BODY>
</HTML>
```

A continuación de explica el código de cambios:

La Figura 3.16 muestra una página Web para el módulo de cambios en el submenú de formación académica.



La Figura 3.16 muestra la página para el módulo de cambios dentro del submódulo de Formación Académica

El código se omite ya que es similar al presentado en la figura 3.12, presentada con anterioridad, se observa en la página las opciones que contiene el submódulo de formación académica y las páginas respectivas a las que se traslada con la obtención de la debida clave del investigador, verificando con ello que existan registros para visualizarlos, en caso contrario desplegará un mensaje que le indique al investigador que no existe información.

La Figura 3.17 muestra la página Web para el rubro de idiomas dentro del módulo de cambios.

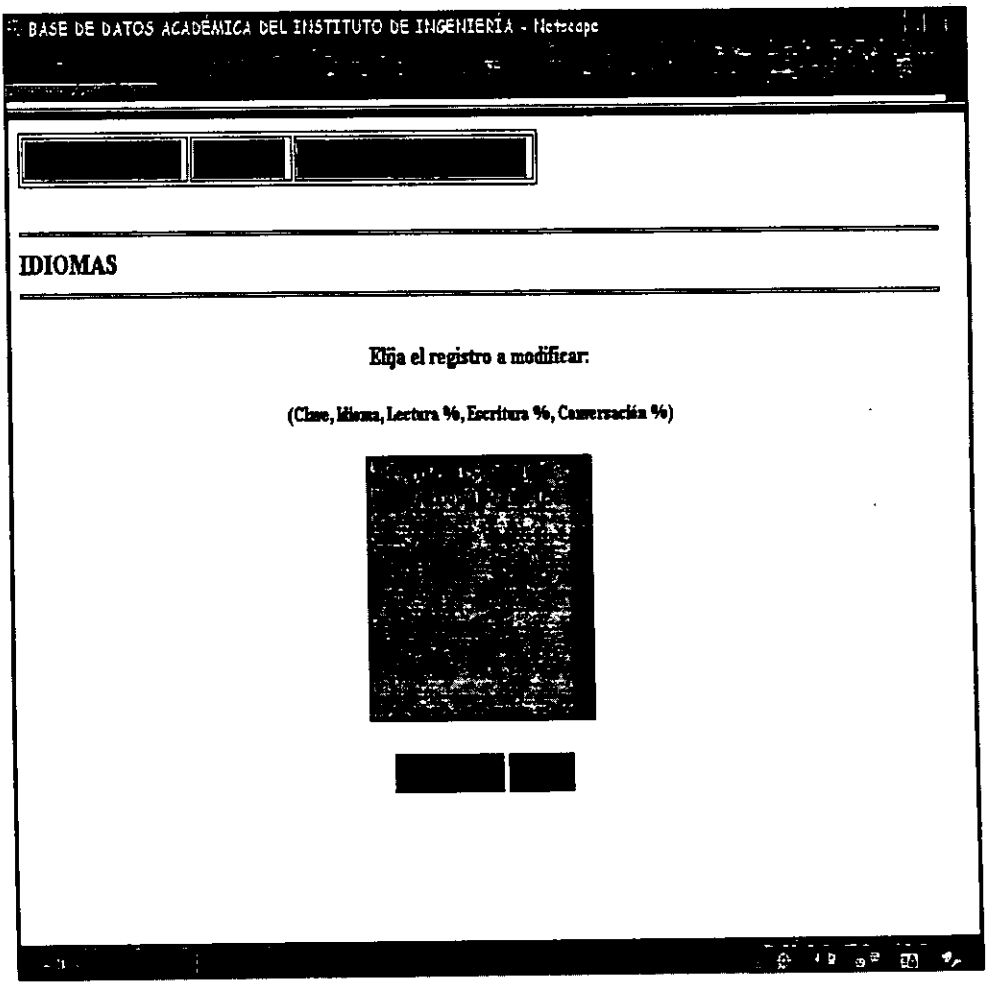


Figura 3.17 muestra la página de idiomas dentro del menú de cambios para el submódulo de Formación Académica.

El código de la página es el siguiente:

Por medio de código en JavaScript, se verifica si se ha seleccionado un registro para continuar a la siguiente página. En caso contrario, despliega un mensaje que le avisa al usuario que no ha seleccionado nada y no le permite avanzar a la siguiente página.

```
<HTML>
<HEAD><TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE></HEAD>
<BODY BGCOLOR=#FFFFFF><P>
<FORM METHOD="POST" ACTION="Idiom.hts" NAME="frm_idiom"
OnSubmit="if(frm_idiom.IDIOMA.selectedIndex<0){alert('Se le olvidó elegir
un registro');return false;}">

<INPUT TYPE="hidden" NAME="var" VALUE="$var" SIZE="10" MAXLENGTH=10>
<TABLE BORDER = "1">
<TR><TH>
<INPUT TYPE="button" NAME="Men&uacute; Principal" VALUE="Men&uacute;
Principal" onClick="history.go(-3)">
</TH>
<TH><INPUT TYPE="button" NAME="menu_cambios" VALUE="Cambios"
onClick="history.go(-2)">
</TH>
<TH>
<INPUT TYPE="button" NAME="menu_forma_acade" VALUE="Formaci&oacute;n
acad&eacute;mica" onClick="history.back()">
</TH>
</TR>
</TABLE>
<H3><HR>IDIOMAS<HR></H3>

<CENTER>
<SYB TYPE=PERL>
if ($var != 0){
  print <<ETI;
<H4>Elija el registro a modificar:</H4>
<H5>(Clave, Idioma, Lectura %, Escritura %, Conversación %)</H5>
<SELECT NAME= "IDIOMA" SIZE=10>
ETI

ws_sql($ws_db, "select distinct
IDIOM_CON.cl_ident,IDIOM_CON.cl_ident,IDIOMAS.des_idioma,IDIOM_CON.lectur
a,IDIOM_CON.escritura,IDIOM_CON.conversa from IDIOM_CON,IDIOMAS where
IDIOM_CON.cl_idioma=IDIOMAS.cl_idioma and
IDIOM_CON.cl_inves=convert(smallint,\"$cl_inves\") order by
IDIOM_CON.cl_ident", "<option value=\"%s\"> %s, %s, %s\%%\, %s\%%\,
%s%%");
print <<ETIQUE;
</SELECT>
```

```
<INPUT TYPE="hidden" NAME="cl_inves" VALUE="$cl_inves">
<BR><BR>
<INPUT TYPE="SUBMIT" VALUE="ACEPTAR">     <INPUT TYPE="button"
NAME="SALIR" VALUE="SALIR" onClick="history.back()">
ETIQUE
}
else{
  print <<ETI;
<H4>No existe informaci&oacute;n actualmente, utilice la opci&oacute;n de
altas del men&uacute; principal para agregar su informaci&oacute;n</H4>
<INPUT TYPE="button" NAME="Regresar" VALUE="REGRESAR"
onClick="history.back()">
ETI
}
</SYB>
</CENTER>
</FORM>
</BODY>
</HTML>
```

La Figura 3.18 muestra la página Web dentro del módulo de cambios para el rubro de idiomas.

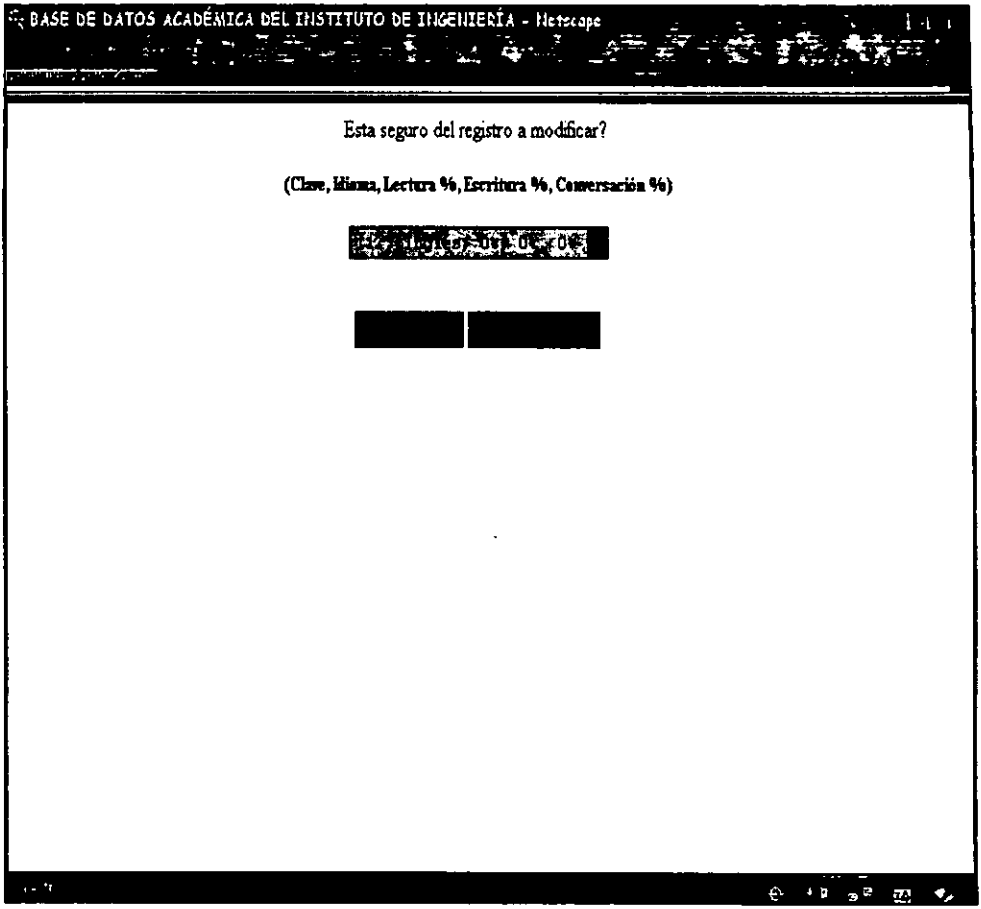


Figura 3.18 muestra la página de idioma dentro del módulo de cambios para el submódulo de Formación Académica.

El código de la siguiente página obtiene la clave del idioma y la almacena en una variable a través de la cual obtiene la información del investigador en particular.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE>
</HEAD>
```

```
</HEAD>
<BODY BGCOLOR=#FFFFFF><P>
<CENTER>Esta seguro del registro a modificar?
<FORM METHOD="POST" ACTION="CA_Idiom.hts">
<SYB TYPE=PERL>
ws_sql($ws_db, "select cl_ident from IDIOM_CON where
cl_ident=convert(smallint, \"$IDIOMA\")", "<INPUT TYPE=\"hidden\"
NAME=\"cl_ident\" VALUE=\"%s\" SIZE=\"10\" MAXLENGTH=\"10\">");
</SYB>
<PRE>
<H5>(Clave, Idioma, Lectura %, Escritura %, Conversación %)</H5>
<SELECT NAME="IDIOM">
<SYB TYPE=PERL>
ws_sql($ws_db, "select distinct
IDIOM_CON.cl_ident,IDIOM_CON.cl_ident,IDIOMAS.des_idioma,IDIOM_CON.lectur
a,IDIOM_CON.escritura,IDIOM_CON.conversa from IDIOM_CON,IDIOMAS where
IDIOM_CON.cl_idioma=IDIOMAS.cl_idioma and
IDIOM_CON.cl_ident=convert(smallint, \"$IDIOMA\")", "<option
value=\"%s\"> %s, %s, %s\%%\, %s\%%\, %s%%");
</SYB>
</SELECT>
</PRE><BR>
<INPUT TYPE="hidden" name="IDIOMA" value="$IDIOMA" SIZE="10"
MAXLENGTH="10">
<INPUT TYPE="hidden" NAME="cl_inves" VALUE="$cl_inves">
<INPUT TYPE="SUBMIT" VALUE="ACEPTAR" NAME="aceptar">     <INPUT
TYPE="BUTTON" name="BACK" VALUE="REGRESAR" onClick="history.back()">
</CENTER>
</FORM>
</BODY>
</HTML>
```

La Figura 3.19 muestra una página Web para el módulo de cambios dentro del rubro de idiomas.

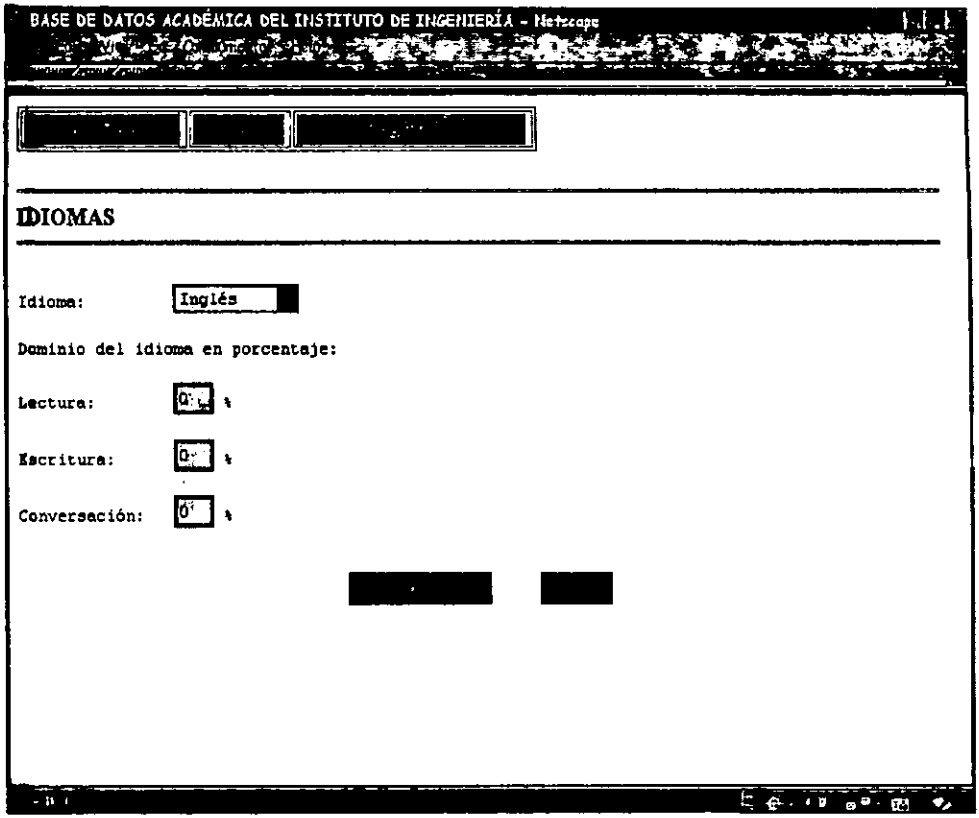


Figura 3.19 muestra la página de idiomas dentro del menú de cambios para el submódulo de Formación Académica.

El código de la siguiente página valida con JavaScript que no se acepten espacios vacíos, desplegando un mensaje cuando esto ocurre.

Se procede a crear cajas de texto en las cuales se despliega la información que previamente se obtuvo a través de una conexión a la base de datos. Se activan los botones en JavaScript para actualizar tanto como para salir.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE>
</HEAD>

<script languaje="JavaScript">
function mensaje(lectura, escritura, conversa){
```

```
if ((lectura == "") || (escritura == "") || (conversa == "")){

       alert("Favor de teclear sus datos.\nNo puede dejar en blanco los
siguientes campos:\nLectura, Escritura, Conversación.");
       document.frm_idioma.lectura.focus();
   }
   else{
if(confirm("Son correctos sus datos?")){
        document.frm_idioma.submit();
        }
   }
}
function valida() {
   if (confirm("Son correctos sus datos?")){
      document.frm_idioma.submit();
   }
}

</script>

<BODY BGCOLOR="white">
<FORM METHOD="POST" ACTION="ACT_CA_Idiom.hts" NAME= "frm_idioma">
<P><P>

<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>
<TABLE BORDER = "1">
<TR><TH>
<INPUT TYPE="button" NAME="menu_bdaii" VALUE="Men&uacute; Principal"
onClick="history.go(-5)">
</TH>
<TH>
<INPUT TYPE="button" NAME="menu_cambios" VALUE="Cambios"
onClick="history.go(-4)">
</TH>
<TH>
<INPUT TYPE="button" NAME="menu_forma_acade" VALUE="Formaci&oacute;n
acad&eacute;mica" onClick="history.go(-3)">
</TH></TR>
</TABLE>
<H3><HR>IDIOMAS<HR></H3>
<PRE>
<SYB TYPE=PERL>
$sql_stmt = qq!select cl_idioma from IDIOM_CON where
cl_ident=convert(smallint,\"$IDIOMA\")!;
if (( $rc = ct_sql( $ws_db, $sql_stmt )) != CS_SUCCEED ) {
       ws_error ("Unable to process database request.");
}
```

```
while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED) {
   RESULTS: {
        if ($result_type == CS_CMD_DONE) {
       last RESULTS;
        }
       if ($result_type == CS_CMD_FAIL) {
        last RESULTS;
        }
        if ( $result_type == CS_ROW_RESULT ) {
         while (@row = ct_fetch ( $ws_db )) {
         $var=@row[0];
                  }
        last RESULTS;
       }
            }
}
if ($ret == CS_FAIL) {
ct_cancel(CS_CANCEL_ALL);
ws_error("A database connection error occurred");
print "error\n";
}

if ($var eq "AL"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option selected value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}

if ($var eq "CA"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option selected value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "CH"){
print <<ETI;
```

```
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option selected value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "ES"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option selected value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "FR"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option selected value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "IN"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option selected value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués
```

```
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "JP"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option selected value="JP">Japonés
<option value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "PO"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option selected value="PO">Portugués
<option value="RU">Ruso
</SELECT>
ETI
}
if ($var eq "RU"){
print <<ETI;
Idioma:          <SELECT NAME="Idiomas">
<option value="AL">Alemán
<option value="CA">Catalán
<option value="CH">Checo
<option value="ES">Español
<option value="FR">Francés
<option value="IN">Inglés
<option value="JP">Japonés
<option value="PO">Portugués<option selected value="RU">Ruso
</SELECT>
ETI
}
</SYB>

Dominio del idioma en porcentaje:
<SYB TYPE=PERL>
ws_sql($ws_db,"select lectura from IDIOM_CON where
cl_ident=convert(smallint, \"$IDIOMA\")","
```

```
Lectura:        <INPUT TYPE=\"text\" NAME=\"lectura\" VALUE=\"%s\"
SIZE=\"3\" MAXLENGTH=\"3\"> %%");

ws_sql($ws_db,"select escritura from IDIOM_CON where
cl_ident=convert(smallint, \"$IDIOMA\")","
Escritura:      <INPUT TYPE=\"text\" NAME=\"escritura\" VALUE=\"%s\"
SIZE=\"3\" MAXLENGTH=\"3\"> %%");

ws_sql($ws_db,"select conversa from IDIOM_CON where
cl_ident=convert(smallint, \"$IDIOMA\")","
Conversación:   <INPUT TYPE=\"text\" NAME=\"conversa\" VALUE=\"%s\"
SIZE=\"3\" MAXLENGTH=\"3\"> %%");
</SYB>
<INPUT TYPE="hidden" NAME="cl_ident" VALUE="$cl_ident" SIZE="10"
MAXLENGTH=10>
<CENTER><INPUT TYPE="button" VALUE="ACTUALIZAR"
onClick="mensaje(document.frm_idioma.lectura.value,
document.frm_idioma.escritura.value,
document.frm_idioma.conversa.value)">     <INPUT TYPE="button"
VALUE="SALIR" onClick="history.go(-3)"></CENTER>
</PRE>
</FORM>
</BODY>
</HTML>
```

La Figura 3.20 muestra una página Web para una actualización terminada dentro del módulo de cambios para el rubro de idiomas.

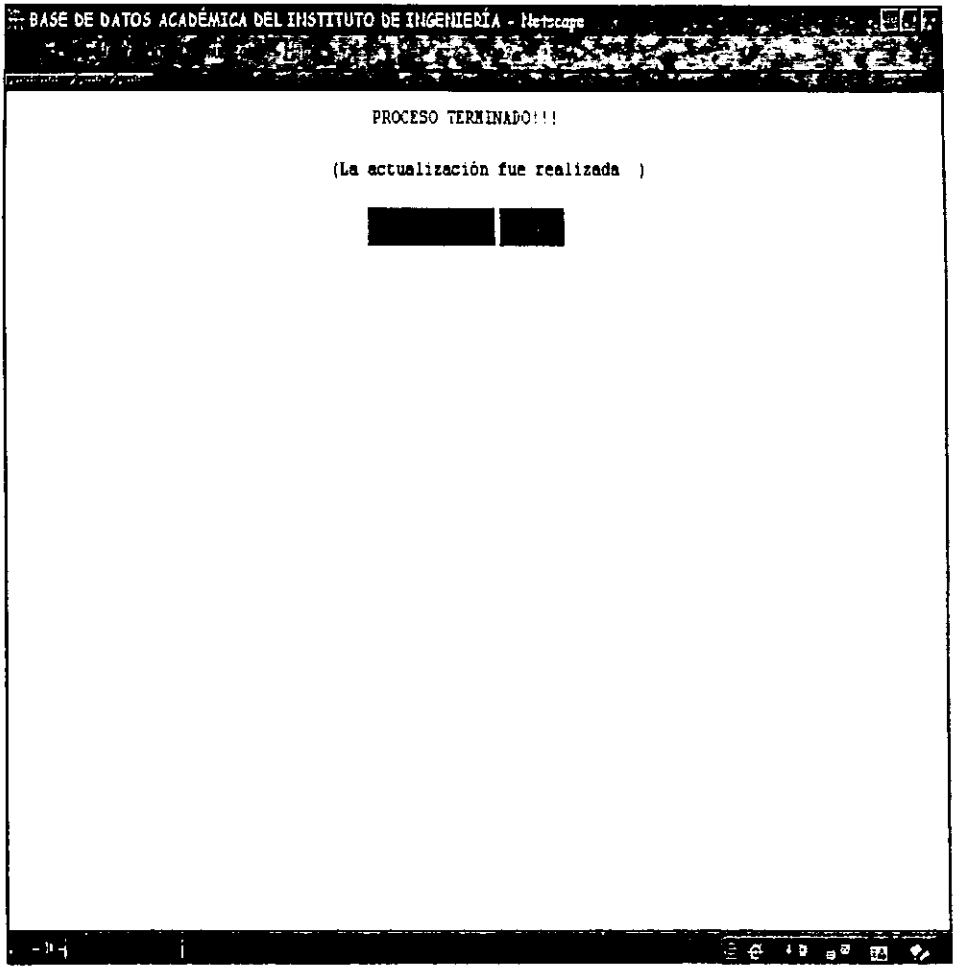


Figura 3.20 muestra una actualización realizada dentro del menú de cambios dentro del submódulo de Formación Académica.

Finalmente en este código se efectúa el proceso de validar toda la información, esto se hace a través de un documento de HTML.

Se cargan todos los valores para verificar la información que se va a actualizar. A su vez se eliminan los espacios en blanco de todos los campos que se crean a través de una caja de texto y se verifica que ningún campo se encuentre vacío.

Una vez validado lo anterior se inicia el procedimiento, preguntando si los campos no son vacíos, si esto se cumple pasa a crear un procedimiento almacenado en la base de datos enviando toda la información que va a actualizar. El código en particular de esta página no se muestra ya que se parece al utilizado en la figura 3.20.

La etapa de cambios, es similar al módulo de bajas. Utilizándose un botón de actualización o uno de eliminación según el caso. Por su parte el módulo de cambios presenta un botón de salir. En este caso se presenta la información resultante de la consulta a través de un archivo plano, sin necesidad de utilizar un procedimiento almacenado asociado a algún botón. A continuación, para el módulo de bajas, cambios y de consultas si se desea consultar algún rubro en el cual no existe información se despliega una página web como la siguiente.

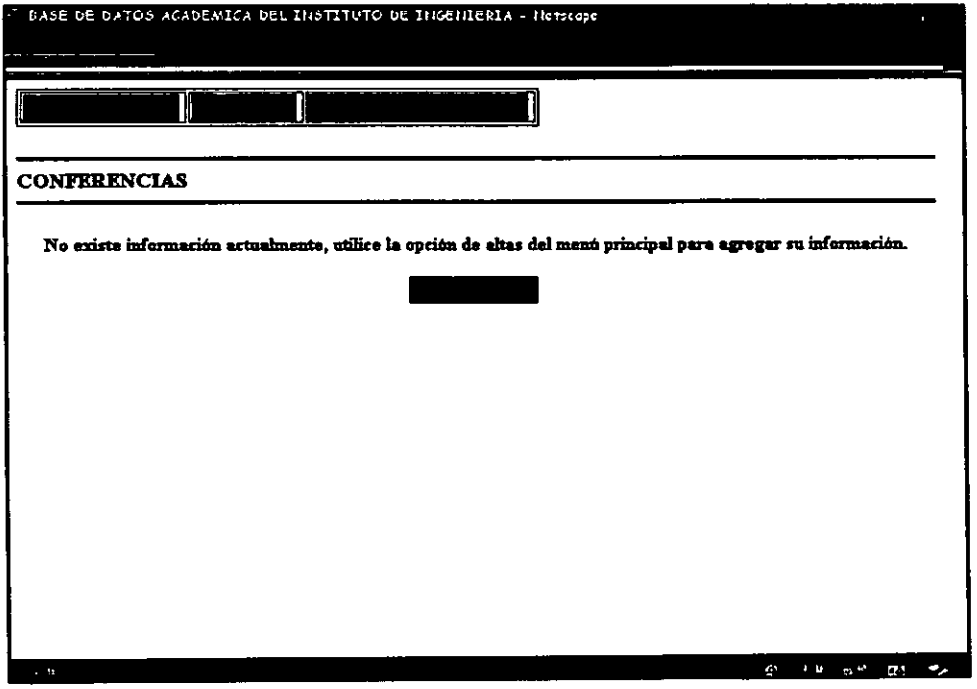


Figura 3.21 Página que se despliega en el momento de no existir información en algún módulo.

El diseño de la página es exactamente igual al manejado en las páginas anteriores, como se observa enseguida.

Lo interesante de esta página es que desde el momento en que se entra al menú principal y se elige el módulo de bajas, cambios o consultas se hace un select del contenido del o los registros de dicha tabla, si está contiene información al llegar a esta página el valor de la variable que arrastra la información es diferente de cero construye un select, de lo contrario envia un mensaje que indica que no se tiene información.

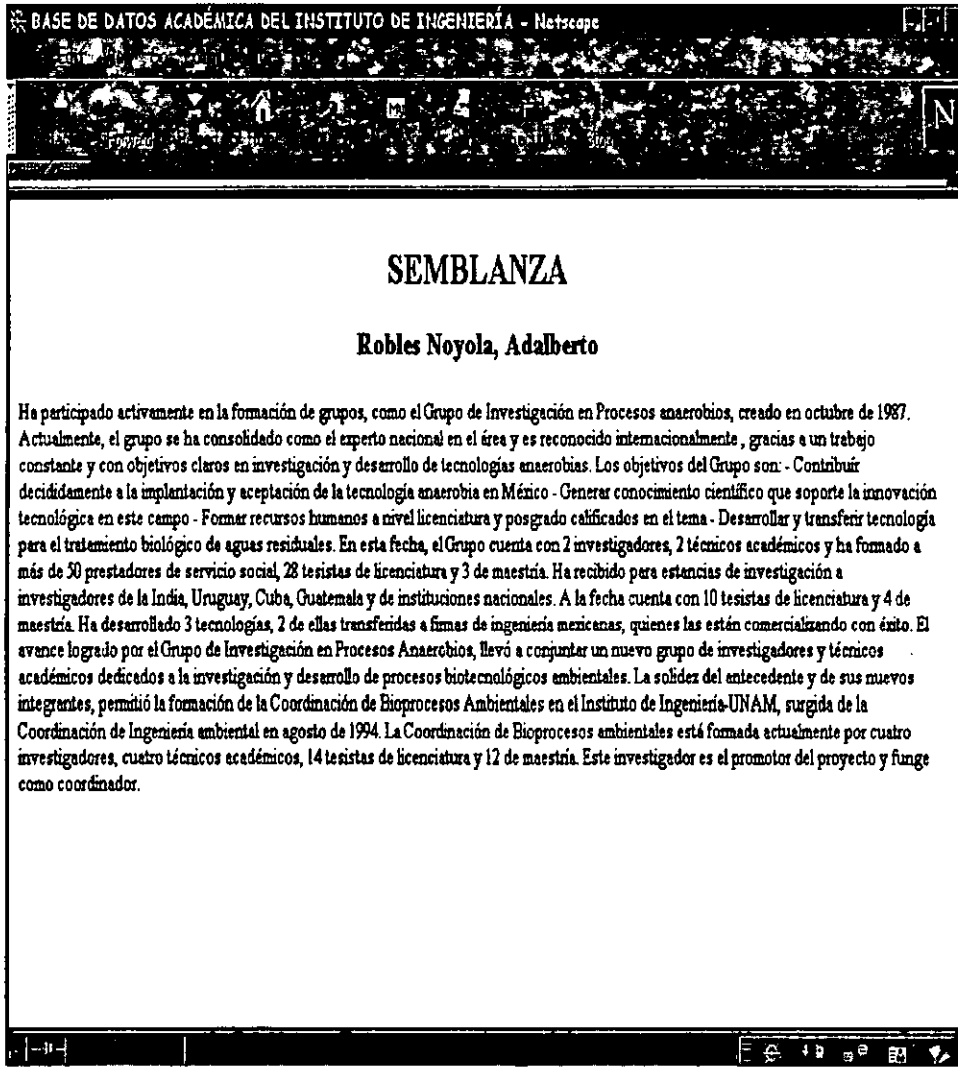
A continuación el código de dicha página:

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE></HEAD>
<BODY  BGCOLOR="white" TEXT="#000000"
LINK="#rrggbb" VLINK="#rrggbb" ALINK="#rrggbb">
<FORM METHOD="POST" ACTION="CO_Actdocco2.hts" NAME="frm_actdocco"
OnSubmit="if(frm_actdocco.CONFIMP.selectedIndex<0){alert('Se le olvidó
elegir un registro');return false;}">
<INPUT TYPE="hidden" NAME="confimp" VALUE="$confimp" SIZE="10"
MAXLENGTH=10>
<TABLE BORDER = "1">
<TR><TH>
<INPUT TYPE="button" NAME="menu_bdaii" VALUE="Men&uacute Principal"
onClick="history.go(-3)">
</TH>
<TH><INPUT TYPE="button" NAME="menu_consultas" VALUE="Consultas"
onClick="history.go(-2)">
</TH>
<TH>
<INPUT TYPE="button" NAME="CO_menu_act_doc" VALUE="Actividades docentes"
onClick="history.back()">
</TH>
</TR>
</TABLE>

<H3><HR>CONFERENCIAS<HR></H3>
<CENTER>
<SYB TYPE=PERL>
if ($confimp != 0){
  print <<ETI;
<H4>Elija la conferencia a consultar.</H4>
<H5>(Clave, Fecha, Título, Evento)</H5>
<SELECT NAME="CONFIMP" SIZE=10>
ETI

ws_sql($ws_db, "select distinct
cl_confimp,cl_confimp,convert(varchar(15),confimp.fecha_even,110),titulo,
```

```
evento from confimp where cl_inves=convert(smallint, \"$cl_inves\") order
by cl_confimp", "<option value=\"%s\">%s, %s, %s, %s");

print <<ETIQUE;
</SELECT>
<INPUT TYPE="hidden" NAME="cl_inves" VALUE="$cl_inves">
<BR><BR>
<INPUT TYPE="SUBMIT" VALUE="ACEPTAR">     <INPUT TYPE="button"
NAME="SALIR" VALUE="SALIR" onClick="history.back()">
ETIQUE
}
else{
  print <<ETI;
<H4>No existe información actualmente, utilice la opción de altas del
menú principal para agregar su información.</H4>
<INPUT TYPE="button" NAME="Regresar" VALUE="REGRESAR"
onClick="history.back()">
ETI
}
</SYB>
</CENTER>
</FORM>
</BODY>
</HTML>
```

Para el módulo de semblanza y de curriculum general se empleará un ejemplo de cómo funcionan y a su vez se explicará el código que los conforman.

La Figura 3.22 representa la opción de semblanza para la BDAII.



La Figura 3.22 muestra la página de semblanza dentro del sistema BDAII

El código de la página se explica a continuación:

Por medio de Perl, se obtiene la clave del investigador y se trae la información que contiene la tabla de semblanza, esto se almacena en diversas variables, si estas no contienen información entonces despliega un mensaje que no exísten datos actualmente. Básicamente este módulo se programó en PERL.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE></HEAD>
<BODY BGCOLOR="white" TEXT="#000000"
LINK="#rrggbb" VLINK="#rrggbb" ALINK="#rrggbb">

<FORM NAME="frm_semblanza" METHOD="POST" ACTION="Semblanza1.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\") ", "<INPUT TYPE=\"hidden\" NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>

<SYB TYPE=PERL>
$tit_grupal="TRUE";
$mensaje="TRUE";

$sql_stmt=qq!select * from SEMBLANZA1 where
cl_inves=convert(smallint, \"$cl_inves\")!;

  if(($rc = ct_sql($ws_db, $sql_stmt)) != CS_SUCCEED){
       ws_error ("Unable to process database request.");
  }
while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED)
 {
   RESULTS:{
             if($result_type == CS_CMD_DONE){
               last RESULTS;
             }
             if($result_type == CS_CMD_FAIL) {
               last RESULTS;
             }
      if($result_type == CS_ROW_RESULT ){
          while (@row = ct_fetch ( $ws_db )) {
                  if($tit_grupal eq "TRUE"){
                print "<P><P><BR>";
                print "<CENTER><H2>SEMBLANZA</H2></CENTER>";
                    print "<P><P>";
                $tit_grupal = "FALSE";
                  }
                    $mensaje="FALSE";
                    print "<BASEFONT size=4><B>";
                    print "<CENTER>";
                    print @row[1];
                    print " ";
                    print @row[2];
                    print ", ";
                   print @row[3];
                    print "</CENTER></FONT></B><BR>";
```

```
                    print "<P><P><P><P>";
                    print "<BASEFONT size=2>";
                          if((length(@row[4])!=1) && (@row[4] ne "")){
                    print @row[4];
                    }
                    print "</FONT>";
                    print "<P><P>";
                   $bandera="T";
}
              last RESULTS;
        }
    }
}
    if($mensaje eq "TRUE"){ #Probando la bandera
       print "<P><P><BR>";
       print "<H2>SEMBLANZA</H2>";
       print "No ex&iacute;sten datos actualmente.";
    }
       print "<P><P>";
     if($ret == CS_FAIL) {
       ct_cancel(CS_CANCEL_ALL);
       ws_error("A database connection error occurred");
       print "error\n";
     }
</SYB>
<BR><BR>
 </FORM>
</BODY>
</HTML>
```

A continuación veremos un ejemplo del formato final de impresión de un curriculum.

CURRÍCULUM VITAE
Castillo Hernández, Gabriel

DATOS PERSONALES

RFC: **********
Lugar de nacimiento: México
Nacionalidad: Mexicana
Estado civil: Casado
Domicilio: **********
Teléfonos: Casa: ******** Oficina: 622-81-39 FAX: 622-81-37
Correo electrónico: gch@pumas.iingen.unam.mx

FORMACIÓN ACADÉMICA

Grados Obtenidos

Licenciatura en Ingeniería en Computación, Facultad de Ingeniería, UNAM, DF, México.
Tesis: "Sistema de diseño de filtros digitales", examen: 08-26-1987.

Otros estudios

Otros

Técnico en procesamiento electrónico de datos, Centro de estudios tecnológicos No. 5, México, ene 01, 1979 a ene 01, 1980
Autocad 2D, DGSCA, UNAM, México, ene 01, 1994 a ene 01, 2001
Autocad 3D, Diseño orientado a objetos, curso propedéutico de la Maestría en Informática, DGSCA, UNAM, México, ene 01, 1994 a ene 01, 2001
Diseño orientado a objetos, Curso propedéutico de la Maestría en Informática, México, ene 01, 2001 a ene 01, 2001
Lenguaje de Programación, México, ene 01, 2001 a ene 01, 2001
Sistemas empleados (Con sus respectivos sistemas operativos), México, ene 01, 2001 a ene 01, 2001
Microcomputradoras, México, ene 01, 2001 a ene 01, 2001
Estaciones de trabajo, México, ene 01, 2001 a ene 01, 2001

Congresos y similares

Nacionales

Software controlador para una red anillo para computadoras personales , México, DF, México, 01 ene 1996 a 01 ene 1996.
Sistema de captura y simulación de una red de distribución de agua potable en ambiente Windows , México, DF, México, 01 ene 1996 a 01 ene 1996.
Desarrollo de un sistema tutorial para apoyo a la materia de Redes de Computadoras , México, DF, México, 01 ene 1996 a 01 ene 1996.
Sistema para el cálculo del fenómeno de oscilación de masa en acueductos , México, DF, México, 01 ene 1996 a 01 ene 1996.
Desarrollo de una biblioteca de funciones en ensamblador , México, DF, México, 01 ene 1996 a 01 ene 1996.
Seminario en la facultad de Ingeniería , México, DF, México, 01 ene 2001 a 01 ene 2001.

CARGOS DESEMPEÑADOS

Académicos

Director de cursos de seminario, ene 2001 a ene 2001.

ACTIVIDADES DOCENTES

Cátedras

Licenciatura

Programación estructurada y características del lenguaje, Facultad de Ingeniería, UNAM, México, ene 1988 a ene 1990.
Programación estructurada y características del lenguaje", Facultad de Ingeniería, UNAM, México, ene 1988 a feb 1990.
Programación de sistemas, Facultad de Ingeniería, UNAM, México, ene 1991 a ene 1996.
"Programación de sistemas", Facultad de Ingeniería, UNAM, México, ene 1991 a ene 2001.

Otros cursos impartidos

Cursos cortos

Software controlador para una red anillopara computadoras personales, México, ene 01, 2001 a ene 01, 2001.
Sistema de captura y simulación de una red de distribución de agua potable en ambiente Windows, México, ene 01, 2001 a ene 01, 2001.
Desarrollo de un sistema tutorial para apoyo a la materia de Redes de Computadoras, México, ene 01, 2001 a ene 01, 2001.
Sistema para el cñalculo del fénomeno de oscilación de masa en acuerdos, México, ene 01, 2001 a ene 01, 2001.
Desarrollo de una biblioteca de funciones en ensamblador, México, ene 01, 2001 a ene 01, 2001.

Tesis dirigidas

Licenciatura

Enrique Ordoñez Angeles (ene 1990) Sistema de despliegue gráfico de la Central Geotérmica Cerro Prieto, *Facultad de Ingeniería*, México.
Enrique Ordoñez Ángeles (ene 1990) "Sistema de despliegue gráfico de la Central Geotérmica Cerro Prieto", *Facultad de Ingeniería*, México.
Emilio Ruiz Galicia (ene 1992) Desarrollo de una tarjeta de conversión analógico digital para una computadora THOSIBA 3200H, *Facultad de Ingeniería*, México.
Marcos Hernández Martínez (ene 1993) Modelado matemático de la red de agua potable de Guadalajara, *Facultad de Ingeniería*, México.
Arturo Becerril, José Román Fonseca Méndez (ene 1994) Análisis, diseño e implementación de un sistema no interactivo de evaluación de pruebas psicométricas para la dirección general de orientación vocacional, *Facultad de Ingeniería*, México.
Arturo Becerril Osnaya, José Roman Fonseca Méndez. (ene 1994) "Análisis, diseño e implementación de un sistema no interactivo de evaluación de pueba psicométricas para la dirección general de orientación vocacional", *Facultad de Ingeniería*, México.
José Bogar Pérez Gutiérrez, Dora Alejandra Tipacamu Ríos (ene 1995) Software controlador de una red de anillo para PC"s, *Facultad de Ingeniería*, México.
Ignacio Caballero Rosas, Eduardo López Castellanos (ene 1995) Sistema de captura y presentación gráfica en ambiente Windows, *Facultad de Ingeniería*, México.
José Borga Pérez Gutiérrez y Dora Alejandra Tipacamu Rios (ene 1995) "Software controlador de una red de anillo para PC's", *Facultad de Ingeniería*, México.
Ignacio Caballero Rosas y Eduardo López Castellanos (ene 1995) "Sistema de captura y presentación gráfica en ambiente Windows", *Facultad de Ingeniería*, México.
Patricia del Valle Morales (ene 1995) Desarrollo de un sistema gráfico de captura para redes de tanques y pozos utiizando la Tecnología Orientada a Objetos, *Facultad de Ingeniería*, México.
Patricia del Valle Morales (ene 1995) "Desarollo de un sistema gráfico de captura par redes de tanques y pozos utilizando la Tecnología Orientada a Objetos", *Facultad de Ingeniería*, México.
Jacqueline Flores Piña (ene 1996) Sistema interactivo multimedia para niños hipoacústicos, *Facultad de Ingeniería*, México.
Jacqueline Flores Piña. (ene 1996) "Sistema interactivo miltimedia para niños hipoacústicos", *Facultad de Ingeniería*, México.
Martín Bravo Aguirre (ene 1996) Programación Orientada a Objetos, un caso práctico: Sistema de Captura de redes de agua potable, *Facultad de Ingeniería*, México.
Martín Bravo Aguirre (ene 1996) "Programación Orientada a Objetos, un caso práctico:Sistema de Captura de redes de agua potable", *Facultad de Ingeniería*, México.

Maestría

Patricia del Valle Morales (ene 1996) Desarrollo de un istema de captura gráfica utilizando Tecnología Orientada a Objetos, *Facultad de Ingeniería*, México.

Tutorías

Licenciatura

Emilio Ruiz Galicia, Facultad de Ingeniería, UNAM, 01-01-1990 a 01-01-1992.
Marcos Hernández Martínez, Facultad de Ingeniería, UNAM, 01-01-1992 a 01-01-1993.

Maestría

Patricia del Valle Morales, Facultad de Ingeniería, UNAM, 01-01-1994 a 01-01-1996.

PUBLICACIONES

Artículos en revistas

Con arbitraje internacional

Castillo Hernández Gabriel, (2001) Informática express,

Con arbitraje nacional

Carmona R., Castillo G., Guevara Y. (1989) "SISTEMA AUTOMATIZADO DE ADQUISICION DE DATOS (SAAD)", *Ingeniería*, editado por Facultad de Ingeniería, UNAM, **LIX**
Carmona R., Castillo G., Guevara Y. (1989) "Sistema automatizado de adquisición de datos (SAAD)", *Facultad de Ingeniería, UNAM*, editado por Ingeniería, **LIX**
Castillo G., (1995) "DESARROLLO DE UN MICROKERNEL PARA COMUNICACION ENTRE OBJETOS EN C ++", *Soluciones Avanzadas, México*, (26)
Castillo G., (1995) "Desarrollo de un microkernel para comunicación entre objetos en C++", *Soluciones Avanzadas*, (26)

Sin arbitraje

Carmona R., Castillo G. (1988) "MANUAL DEL USUARIO DEL SISTEMA DE GRAFICACION PCGRAFIC", *Serie Azul*, editado por Instituto de Ingeniería, UNAM, (510) 33.

Informes

Sánchez H., Carmona R., Castillo G. (1987) "Análisis del funcionamiento hidráulico de la válvula de flotador APCO-MEX de 4", *Secretaría de Agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 18, México.
Sánchez H., Carmona R., Castillo G. (1987) Análisis del funcionamiento hidráulico de la válvula de flotador APCO-MEX de 4 pulgadas, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.
Carmona R., Solorio A., Sánchez A., Carmona L., Castillo G., Sánchez J. (1987) Control de sobrepresiones en el Acueducto Río Uspanapa-La Cangrejera utilizando válvulas de alivio, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.
Carmona R., Solorio A., Sánchez A., Carmona L., Castillo G., Sánchez J. (1987) "Control de sobrepresiones en el Acueducto Río Uspanapa - La Cangrejera utilizado válvulas de alivio", *Secretaría de Agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 27, México.
Aguilar L., Carmona R., Sánchez J.L., Castillo G., Sánchez A., Carmona L., Rodal E., Solorio A., Guevara Y. (1987) Estudio teórico-experimental del transitorio hidráulico en instalaciones de bombeo en pozo para abastecimiento de agua, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.
Aguilar L., carmona R., Sánchez J.L., Castillo G., Sánchez A., Carmona L., Rodal E., Solorio A., Guevara Y. (1987) "Estudio teórico-experimental del transitorio hidráulico en instalaciones de bombeo en pozos para abastecimiento de agua", *Secretaría de Agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 150, México.
Carmona R., Castillo G. (1988) "Manual del usuario del sistema de graficación PCGRAFIC", *Instituto de Ingeniería*, UNAM, Serie azul del Instituto de Ingeniería, proy 510, pág 33, México.
Carmona R., Castillo G., Aguilar L. (1988) Revisión del diseño hidráulico del acueducto San Francisquito-Mazatlán, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.
Carmona R., Castillo G., Aguilar L. (1988) "Revisión del diseño hidráulico del acuerdo San Francisco-Mazatlan", *Secretaría de Agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 67, México.
Carmona R., Sánchez A., Aguilar L., Castillo G. (1988) "Informe de la campaña de mediciones al sistema de agua de circulación de la unidad #2 de la C.T. Manzanillo II", *Comisión Federal de Electricidad*, Realizado para: Comisión federal de Electricidad, pág 41, México.
Carmona R., Sánchez A., Aguilar L., Castillo G. (1988) Informe de la campaña de mediciones al sistema de agua de circulación de la unidad no. 2 de la C.T. Manzanillo II, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Federal de Electricidad, pág , México.

Aguilar L., Carmona R., Rodal E., Castillo G. (1988) Verificación en prototipo del punto de operación de mayor eficiencia en las bombas del Acueducto Chapala-Guadalajara, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.

Aguilar L., carmona R., Rodal E., Castillo G. (1988) "Verificación en prototipo delpunto de operación de mayor eficiencia en las bombas del Acueducto Chapala-Guadalajara", *Secretaría de Agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 16, México.

Sánchez A., Carmona R., Rodal E., Guevara Y., Castillo G., Sámano A. (1988) Informe de los resultados obtenidos de la campaña de medición en la Planta de Bombeo No. 2 del Sistema Cutzamala, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.

Aguilar L., Carmona R., Castillo G., Sánchez A., Rodal E. (1988) Resultados de las mediciones del transitorio hidráulico en la red de pozos del acueducto Valle del Verano-Hidalgo del Parral, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.

Sánchez A., Carmona R., Castillo G., Guevara Y., Valencia J., Sámano A. (1988) "Informe de los resultados obtenidos en las campañas de medición de las plantas de bombeo No's 5, 5A, y del Sistema Cutzamala", *Comisión de Aguas del Valle de México*, Realizado para: Comisión de Aguas del Valle de México, pág 61, México.

Sánchez A., Carmona R., Castillo G., Guevara Y., Valencia J., Sámano A. (1988) Informe de los resultados objetivos en las campañas de medición de las plantas de bombeo No. 5, 5A y 3 del Sistema Cutzamala, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión de Aguas del Valle de México, pág , México.

Sánchez A., Carmona R., Rodal E., Guevara Y., Castillo G. Sámaro A. (1988) "Informe de los resultados obtenidos de la campaña de medición en la Planta de Bombeo #2 del Sistema Cutzamala", *Secretaría de Agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 41, México.

Aguilar L., carmona R., Castillo G., Sánchez A., Rodal E. (1988) "Resultados de las mediciones del transitorio hodráulico en la red de pozos del acueducto Valle del verano - Hidalgo de Parral", *Secretaría de Agricultura y Recursos Hidraulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidraulicos, pág 39, México.

Carmona R., Sánchez A., Guevara Y., Castillo G., Rodal E., Sámano A. (1988) Informe de resultados de la campaña de mediciones en el Acueducto Río Colorado-Tijuana los días 20 al 25 de septiembre de 1988, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.

Carmona R., Sánchez A., Guevara Y., Castillo G., Rodal E., Sámano A. (1988) "informe de resultados de la campaña de mediciónes en el Acueducto Río Colorado-Tijuana los días 20 al 25 de septiembre de 1988", *Secretaría de Agricultura y Recursos Hidraúlicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 43, México.

Castillo G. et. al., (1989) "informe de la campaña de mediciones del Acueducto San Francisco-Mazatlán efectuada los días 12 al 15 de abril de 1989", *Comisión Nacional del Agua*, Realizado para: Comisión Nacional del Agua, pág 78, México.

Castillo G., (1989) Informe de la campaña de mediciones del Acueducto San Francisquito-Mazatlán efectuada los días 12 al 15 de abril de 1989, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.

Aguilar L., Carmona R., Rodal E., Castillo G., Guevara Y. (1989) "Análisis de las funciones de presión en la succión de las bombas del Acueducto Río Colorado-Tijuana", *Comisión Nacional del Agua*, Realizado para: Comisión Nacional del Agua, pág 16, México.

Sánchez A., Carmona R., Castillo G., Carmona L. (1989) Análisis del transitorio hidráulico en el tramo PB4-TS5 del Acueducto Río Colorado-Tijuana, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Secretaría de Agricultura y Recursos Hidráulicos, pág , México.

Aguilar L., Carmona R., Rodal E., Castillo G., Guevara Y (1989) Análisis de las fluctuaciones de presión en la succión de las bombas del Acueducto Río Colorado-Tijuana, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.

Sánchez A., Carmona R., Castillo G., Carmona L. (1989) "Análisis del Transitorio hidráulico en el tramo PB4-TS5 del acueducto Río Colorado-Tijuana", *Secretaría de agricultura y Recursos Hidráulicos*, Realizado para: Secretaría de Agricultura y Recursos Hidráulicos, pág 28, México.

Castillo G., (1989) Revisión del funcionamiento hidráulico y diseño del sistema de control de transitorios del Acueducto San Francisco del Rincón-León, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para el Sistema de Agua Potable y Alcantarillado del Municipio de León, Guanajuato, pág , México.

Castillo G. et. al., (1989) "Informe de la campaña de mediciones para la determinación del coeficiente de fricción Darcy-Weisbach en la línea de conducción del Acueducto Linares-Monterrey, efectuada los días 7,8 y 9 de septiembr, *Comisión Nacional del Agua*, Realizado para: Comisión Nacional del Agua, pág 21, México.

Castillo G., (1989) Informe de la campaña de mediciones para la determinación del coeficiente de fricción Darcy-Weisbach en la línea de conducción del Acueducto Linares-Monterrey efectuada 7,8 y 9 de sept. de 1989, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.

Castillo G. et. al., (1989) "Revisión del funcionamiento hidráulico y diseño del sistema de control de transitorios del Acueducto San Francisco del Rincon-León"", *Sistema de Agua Potable Y Alcantarillado del Municipio de León, Gto.*, Realizado para: Sistema de Agua Potable y Alcantarillado del Municipio de León, Gto., pág 37, México.
Rodal E., Carmona R., Aguilar L., Guevara Y., Castillo G., Sámano A. (1989) "Determinación del funcionamiento de las turbinas de la C.H. Manuel Moreno Torres al operar a carga parcial sin aletas en sus desfogues.Medición en la U-3, mayo 1989", *Comisión Federal de Electricidad*, Realizado para: Comisión Federal de Electricidad, pág 59, México.
Rodal E., Carmona R., Aguilar L., Guevara Y., Castillo G., Sámano A. (1989) Determinación del funcionamiento de las turbinas de la C.H. Manuel Moreno Torres al operar a carga parcial sin aletas en sus desfogues. Mediciones en la U-3, mayo 1989, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Federal de Electricidad, pág , México.
Castillo G. et. al., (1990) "Puesta en operación del equipo del laboratorio de modelos hidráulicos y fluviales de la Comisión Nacional del Agua (Primera Parte)", *Comisión Nacional del Agua*, Realizado para: Comisión Nacional del Agua, pág 12, México.
Castillo G., (1990) Puesta en operación del equipo del laboratorio de modelos hidráulicos y fluviales de la Comisión Nacional del Agua (Primera Parte), *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.
Cruickshank C., Ordoñes E., Castillo G. (1990) Modelo matemático de la red de vaporductos de la planta Geotermoeléctrica de Cerro Prieto, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Gerencia de Plantas Geotermoeléctricas de la CFE, pág , México.
Cruickshank C., Ordoñes E., Castillo G. (1990) "Modelo matemático de la red de vaporductos de la planta Geotermoeléctrica de Cero Prieto", *Gerencia de Plantas Geotermoeléctricas de la CFE*, Realizado para: Gerencia de Plantas Geotermoeléctricas de la CFE, pág 75,,México.
Carmona R., Sánchez A., Castillo G., Sámano A., Carmona G., García J. (1990) Revisión del funcionamiento de la cámara de aire diseñada para el control del transitorio hidráulico en la planta de bombeo Las Amalias del acueducto San Francisco del Rincón-León, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la SAPAL, pág , México.
Carmona R., Sánchez A.,Castillo G., Sámano A., Carmona G., García J. (1990) "Revisión del funcionamiento de la cámara de aire diseñada para el control del transito hidráulico en la planta de bombeo Las Amalias del acueducto San Francisco del Rincón-León", *SAPAL del municipio de León Guanajuato*, Realizado para: SAPAL del municipio de León Guanajuato, pág 17, México.
Carmona R., Lomónaco P., Carmona L., Sánchez A., Castillo G. (1991) "Calibración del método para análisis y diseño de redes de abastecimiento de agua", *Comisión Nacional del agua*, Realizado para: Comisión Nacional del agua, pág 90, México.
Carmona r., Lomónaco P., Carmona L., Sánchez A., Castillo G. (1991) Calibración del método numérico para análisis y diseño de redes de abastecimiento de agua, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.
Carmona L., Carmona G., Castillo G. (1994) "Simulación dinámica de redes de distribución de agua potable", *Comisión Nacional del Agua*, Realizado para: Comisión Nacional del Agua, pág 93, México.
Carmona L., Carmona R., Castillo G. (1994) Simulación dinámica de redes de distribución de agua potable, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.
Castillo G., Del Valle P. (1995) "Diseño e implementación de un sistema de captura gráfica para programas de sistemas hidráulicos basados en la Tecnología Orientada a Objetos", *Informe Interno*, Informe Interno, pág 43, México.
Castillo G., Del Valle P. (1995) Diseño e implementación de un sistema de captura gráfica para programas de sistemas hidráulicos basado en la Tecnología Orientada a Objetos, *Instituto de Ingeniería*, UNAM, Realizado para: Instituto de Ingeniería, pág , México.
Castillo G., Carmona R. (1996) "Análisis de las alternativas de recubrimiento de la tubería del Acueducto Río Colorado Tijuana", *Comisión Nacional del Agua*, Realizado para: Comisión Nacional del Agua, pág 30, México.
Castillo G., Carmona R. (1996) Análisis de las alternativas de recubrimiento de la tubería del Acueducto Río Colorado-Tijuana, *Instituto de Ingeniería*, UNAM, Realizado para: Elaborado para la Comisión Nacional del Agua, pág , México.
Yanet Enríquez Chávez, Gabriel Castillo Hernández (1997) "Manual de HTML 3.0", *Informe del proyecto de control de contratos y proyectos*, Informe del proyecto de control de contratos y proyectos, pág 61, México.

Memorias

Internacionales

Aguilar L., Carmona R., Castillo G., (jul 1988) "Criterio de cálculo del transitorio hidráulico por corte de bombeo en pozos". *Memorias del XIII Congreso Latinoamericano de hidráulica*. 12 La Habana, Cuba.
Cagigas R., Castillo G., (jul 1988) "Criterio para la determinación del perfil de flujo rápidamente variadoen la descarga libre de un canal". *Memorias del XII Congreso Latinoamericano de Hidráulica*. 12 La Habana, Cuba.

Sánchez A., Carmona R., Castillo G., (jun 1990) "Estudio experimental del funcionamiento hidromecánico de dos válvulas de control automático". *XIV Congreso Latinoamericano de Hidráulica*. 12 Montevideo, Uruguay.

DESARROLLOS

Desarrollos tecnológicos

Soporte en laconversión de imágenes entre diversoso formatos., Castillo Hernández Gabriel, Correción de errores de programación en sistemas escritos para estaciones de trabajo , Instituto de Ingeniería, México, ene 2001.
Simulador del comportamiento hidráulico del acueducto Linares-Monterrey, Castillo Hernández Gabriel, , Instituto de Ingeniería, UNAM, México, ene 2001.
Desarrollo de un sistema de graficación (Pcgrafic), interactivo, Castillo Hernández Gabriel, Para apoyo en la interpretación gráfica de resultados experimentales en el laboratorio de Hidromecánica , Instituto de Ingeniería, UNAM, México, ene 2001.
Desarrollo de un sistema de cálculo numérico en redes de pozos y tanques., Castillo Hernández Gabriel, Tanto en flujo estable como en flujo transitorio, Instituto de Ingeniería, UNAM, México, ene 2001.
Desarrollo del software para el manejo de una tarjeta de conversión analógico-digital, Castillo Hernández Gabriel, (Sistema de adquisición de datos en el campo) , Instituto de Ingeniería, UNAM, México, ene 2001.
Pruebas del hardware y desarrollo del software controlador de una tarjeta de conversión, Castillo Hernández Gabriel, analógico-digital desarrollada por el Instituto de Ingeniería., Instituto de Ingeniería, UNAM, México, ene 2001.
Estudio de una tarjeta para comunicaciones entre una PC y diversos instrumentos de laboratorio, Castillo Hernández Gabriel, La comunicación es con base en el protocolo de comunicación IEEE-488 (HP-1B) , Instituto de Ingeniería,UNAM, México, ene 2001.
Revisión del diseño hidráulico del acuerdo San Francisquito-Mazatlán, Castillo Hernández Gabriel, , Instituto de Ingeniería, UNAM, México, ene 2001.
El análisis del transitorio hidráulico en conducciones parcialmente llenas, Castillo Hernández Gabriel, Instituto de Ingeniería, UNAM, México, ene 2001.
Asesorías sobre transitorios hidráulicos, Castillo Hernández Gabriel, En acuerdos a personal de la Secretaría de Agricultura y Recursos Hidráulicos., Instituto de Ingeniería, UNAM, México, ene 2001.
Asesoría y soporte en general sobrecomputación y desarrollo de sistemas, Castillo Hernández Gabriel, Al Grupo de Hidromecánica , Instituto de Ingeniería, México, ene 2001.
Desarrollo del modelo matemático de la red de aguapotable de la Ciudad de Guadalajara, Jalisco, Castillo Hernández Gabriel, Participaciónen diversas campañas de medición de fenómenos hidráulicos, Instituto de Ingeniería, UNAM, México, ene 2001.
Desarrollo de un sistema de graficación sustituto del paquete PCGRAFIC (desarrollado en anteriores), Castillo Hernández Gabriel, Este nuevo paquete ofrece muchas más ventajas que su antecesor. Este paquete se ha denominado FG., Instituto de Ingeniería, UNAM, México, ene 2001.
desarrollo de un modelo numérico para estudios del fenómeno de oscilación de masa, Castillo Hernández Gabriel, Utilizando técnicas orientadas a objetos., Instituto de Ingeniería, UNAM, México, ene 2001.
Desarrollo de las Hojas HTML del informe de Actividades de 1994 del Instituto de Ingeniería, Castillo Hernández Gabriel, Instituto de Ingeniería, UNAM, México, ene 2001.
Calibración numérica del modelo de la operación del Acuerdo Cutzamala, Castillo Hernández Gabriel, Instituto de Ingeniería, UNAM, México, ene 2001.
Aplicación de las técnicas de Intranet en el Instituto de Ingeniería, Castillo Hernández Gabriel, Instituto de Ingeniería, UNAM, México, ene 2001.
Desarrollo de un sistema de anñalisis lexicográfico en C., Castillo Hernández Gabriel, Este sistema se integró a un paquete escrito en CLIPER, TRANTOR SITEMAS S.A., México, ene 2001.
Desarrollo y mantenimiento del sistemas de adquisición de información denominado EASY FLOW, Castillo Hernández Gabriel, Desarrollado como una parte interface del sistema PHOENICS. El sistema en cuestión se desarrolló originalmente en PASCAL para un PC 386., Consultores en Hidrodinámica y Análisis Matemático (CHAM de, México, ene 2001.
Desarrollo de un sistema de cálculo financiero en una HP9000 para la Comisión Nacional Bancaria., Castillo Hernández Gabriel, Este sistema maneja una base de datos con un volumen de 1´000,000 de registro aproximadamente., Felipe Ochoa y Asociados, Consultores (FOA S.A.), México, ene 2001.
Sitema de Graficación PCgrafic, Programa de computadora de próposito General, Castillo Hernández Gabriel, Ejecutable en computadoras personales PC y compatibles. Permite graficar tanto en patalla como en papel (a través de un graficador o una impresora), México, ene 2001.

Sistema de modelado del comportamiento hidráulico en redes de pozos y tanques., Castillo Hernández Gabriel, Programa de computadora de uso específico elaborado para la SARH.Modela Redes de pozos y tanques tanto en funcionamiento establecido como en funcionamiento transitorio por corte de bombeo., México, ene 2001.
Sistema de graficación Fast Graph, con mejores características que el PC-Graphic, Castillo Hernández Gabriel, México, ene 2001.

Fecha de la última actualización: 13/08/1999

El código que conforma el curriculum en general por ser extenso, no se muestra, pero está disponible para su consulta en el Instituto de Ingeniería.

La elaboración del Curriculum general se hizó a través de vistas, las cuales nos proveen un mecanismo de seguridad, como no contienen datos no ocupan espacio ya que son espejos de varias tablas. Se utilizó HTML, Perl y Web.sql, HTML fue básicamente para el diseño de la página, Perl sirvió para que a través de Web.sql con el comando SELECT se obtuviera la información de las tablas, está se almacena en un arreglo del cual se van tomando los diversos valores y lo imprime. Para los títulos se activó una bandera en la que se preguntaba si el valor era verdadero entonces lo mandaba a imprimir incluyendo los subtitulos necesarios, si la bandera era falsa se ignoraba ese tópico. En el caso de las fechas, como el servidor de bases de datos Sybase envía la fecha en el formato mm/dd/aaaa (en el caso específico del mes lo envía en inglés), se utilizó la función Split de Perl para convertir los diferentes meses a español.

3.3 Validación

En el desarrollo de las páginas se procedió a validar mucha de la información suministrada por el usuario con el fin de que lo que se almacenara en la base de datos fuera correcto y de tal forma que no se presentara ningún error por parte del servidor, por lo cual se trató de hacerle saber al usuario algunos formatos que debía respetar.

En la página principal existe un bloque programado en JavaScript, a través del cual se solicita al usuario proporcione un login y password evitando que se deje algún dato vacío que afecte a la base de datos. En caso de que el usuario no proporcione ningún dato se despliega un letrero que le indica "Favor de teclear sus datos", una vez que estos han sido suministrados se le pregunta si son correctos sus datos, tal y como se muestra a continuación:

```
<script languaje="JavaScript">
function mensaje(u_name, passw){
     if ((u_name == "") || (passw == "")) {
         alert("Favor de teclear sus datos");
         document.forma.u_name.focus();
     }
     else{
         if(confirm("Tecleo correctamente su login y password?")){
         document.forma.submit();
         }
     }
```

```
function valida() {
     if (confirm("Son correctos tus datos?")){
     document.forma.submit();
     }
}
</script>

<FORM METHOD="POST" ACTION="menu_bdaii.hts" NAME="forma"><BR><BR><P>
<CENTER>
<PRE>Login: <INPUT TYPE="TEXT" NAME="u_name" SIZE="8"
MAXLENGTH=15></PRE><BR><P>
<PRE>Password:  <INPUT TYPE="password" NAME="passw" SIZE="8"
MAXLENGTH=15></PRE><BR><P>
<INPUT TYPE="button" VALUE="ACEPTAR" NAME="aceptar"
onClick="mensaje(document.forma.u_name.value,
document.forma.passw.value)">
</CENTER>
</FORM>
```

Una vez que el usuario ha proporcionado sus datos, se utiliza la función de ws_sql para realizar una consulta a la base de datos y verificar que el login y password suministrados hayan sido los correctos y de esta manera permitirle o no al usuario entrar a ver los módulos que integran la BDAII. En caso de ser incorrectos se despliega un letrero que le indica que falló el login y que por lo tanto no puede accesar al sistema.

```
<SYB TYPE=PERL>
$SRV="SYBASE11";
$ws_db=ct_connect($u_name, $passw, $SRV);

if ($ws_db eq "" ) {
    print <<ETI;

      <FONT COLOR="FF0000">
      <H3><CENTER>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
      INGENIER&Iacute;A</CENTER></H3>
      <CENTER><INPUT TYPE="button" VALUE="REGRESAR" NAME="salir"
      onClick="history.back()"></CENTER>
      </FONT>
    ETI
      ws_error ("Fall&oacute; el login.");
}
else{
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\")",  "<INPUT  TYPE=\"hidden\"  NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");

ws_sql($ws_db, "select cl_inves from dat_person2 where u_name=\"$u_name\"
",  "<INPUT  TYPE=\"hidden\"  NAME=\"cl_inves\"  VALUE=\"%s\"  SIZE=\"5\"
MAXLENGTH=\"5\">");
print <<ETI;
</SYB>
}
```

La siguiente validación se realizó con el fin de saber que existiera información en la parte de bajas, cambios o consultas que deseara realizar un investigador. De tal forma que primero se procedió a checar que al menos existiera un registro por medio del siguiente query:

```
<FORM METHOD="POST" action="BA_Cardes1.hts">
<SYB TYPE = PERL>
ws_sql($ws_db, "select cl_inves from dat_person2 where cl_inves= convert
(smallint,\"$cl_inves\")  ",  "<INPUT  TYPE=\"hidden\"  NAME=\"cl_inves\"
VALUE=\"%s\" SIZE=\"5\" MAXLENGTH=\"5\">");
</SYB>

<SYB TYPE=PERL>
$sql_stmt=qq!select cl_cargo from cargos where
cl_inves=convert(smallint,\"$cl_inves\")!;
   if (( $rc = ct_sql($ws_db, $sql_stmt )) != CS_SUCCEED ) {
      ws_error ("No encuentro datos.");
   }
   while ((($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED) {
      RESULTS: {

                  if ($result_type == CS_CMD_DONE) {
                        print "Comando ejecutado"."<p>";
                        last RESULTS;
               }
               if ($result_type == CS_CMD_FAIL) {
             print "HOLA";
                        last RESULTS;
               }

               if ($result_type == CS_ROW_RESULT ) {
                    while (@row = ct_fetch( $ws_db )) {
                        $cargos=@row[0];
                       $mensaje="FALSE";
                     } # Fin del while
                  last RESULTS;
                } # fin del if
              } # fin del Bloque RESULTS
      } # fin del while

   if ($ret==CS_FAIL){
      ct_cancel(CS_CANCEL_ALL);
      ws_error("Error"),
   }
</SYB>
<INPUT TYPE="hidden" NAME="cargos" VALUE="$cargos " SIZE="10"
MAXLENGTH=10>
<LI>
<INPUT TYPE="SUBMIT" VALUE="Cargos desempe&ntilde;ados"
NAME="cargos_desem"></LI>
</FORM>
```

Una vez realizado dicho query se guarda en una variable el número de registros almacenados y posteriormente se pregunta si al menos existe un registro, en caso de que esto ocurra se despliega la información relacionada con este. En caso contrario se despliega un letrero al investigador en el cual se le indica que no existe información en la opción que el desea dar de baja, actualizar o simplemente consultar.

Por otra parte, en caso de que exista más de un registro de información se despliega una lista que le brinda al usuario el poder seleccionar el registro que desea dar de baja, actualizar o consultar, en caso de que el usuario no seleccione nada y de un click en el botón de aceptar se despliega un letrero en el que se le indica que se le olvidó elegir el registro que desea dar de baja, actualizar o consultar y no se pasa a la siguiente página hasta que no seleccione alguno.

```
<HTML>
<HEAD>
<TITLE>BASE DE DATOS ACAD&Eacute;MICA DEL INSTITUTO DE
INGENIER&Iacute;A</TITLE></HEAD>
<BODY  BGCOLOR="white" TEXT="#000000"
LINK="#rrggbb" VLINK="#rrggbb" ALINK="#rrggbb">
<FORM METHOD="POST" ACTION="BA_Cardes2.hts" NAME="frm_cardes"
OnSubmit="if(frm_cardes.CARGOS_DESEM.selectedIndex<0){alert('Se le olvidó
elegir un registro');return false;}">
<INPUT TYPE="hidden" NAME="cargos" VALUE="$cargos" SIZE="10"
MAXLENGTH=10>
<TABLE BORDER = "1">
<TR><TH>
<INPUT TYPE="button" NAME="menu_bdaii" VALUE="Men&uacute; Principal"
onClick="history.go(-2)">
</TH>
<TH><INPUT TYPE="button" NAME="menu_bajas" VALUE="Bajas"
onClick="history.back()">
</TH>
</TR>
</TABLE>
<H3><HR>CARGOS DESEMPEÑADOS<HR></H3>
<SYB TYPE=PERL>
if ($cargos != 0){
  print <<ETI;
<H4>Elija el cargo a eliminar.</H4>
<H5>(Clave, Fecha, Nombramiento, Institución)</H5>
<SELECT NAME="CARGOS_DESEM" SIZE=10>
ETI

ws_sql($ws_db, "select distinct
cl_cargo,cl_cargo,convert(varchar(15),cargos.f_in_tra,110),nomb,instituc
from cargos where cl_inves=convert(smallint,\"$cl_inves\") order by
cl_cargo", "<option value=\"%s\"> %s, %s, %s, %s, %s");
```

```
      print <<ETIQUE;
</SELECT>
<INPUT TYPE="hidden" NAME="cl_inves" VALUE="$cl_inves">
<BR><BR>
<INPUT   TYPE="SUBMIT"   VALUE="ACEPTAR">            <INPUT   TYPE="button"
NAME="SALIR" VALUE="SALIR" onClick="history.back()">
ETIQUE
}
else{
  print <<ETI;
<H4>No existe informaci&oacute;n actualmente, utilice la opci&oacute;n de
altas del men&uacute; principal para agregar su informaci&oacute;n</H4>
<INPUT TYPE="button" NAME="Regresar" VALUE="REGRESAR"
onClick="history.back()">
ETI
}
</SYB>
</CENTER>
</FORM>
</BODY>
</HTML>
```

Existía la discrepancia de que algunas veces los investigadores suministraban comillas (") y otra veces apóstrofe ('). Por lo que se optó que no importando lo que suministraran se hiciera el cambio de comillas (") a apóstrofe (').

A su vez todos los campos que aparecían como NULL en la base de datos se cambian a un " " (espacio en blanco) por medio de la sentencia isnull que se encuentra dentro de la sentencia sql.

```
<SYB TYPE = PERL>
$sql_stmt=qq!select isnull(nom_carr, " ") from est_prof where
cl_est_prof=convert(numeric, \"$GRADO\")!;

  if(($rc = ct_sql($ws_db, $sql_stmt)) != CS_SUCCEED){
       ws_error ("Unable to process database request.");
  }
while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED)

{
   RESULTS:{
             if($result_type == CS_CMD_DONE){
               last RESULTS;
             }
             if($result_type == CS_CMD_FAIL) {
               last RESULTS;
             }
```

```
if($result_type == CS_ROW_RESULT ){
          while (@row = ct_fetch ( $ws_db )) {
                        $var1=@row[0];

$var1 =~ s/(\")+/\'/g;
                        $mensaje="FALSE";
           }#while
             last RESULTS;
       }#if
   }#RESULT
}#while
      if($ret == CS_FAIL) {
      ct_cancel(CS_CANCEL_ALL);
      ws_error("A database connection error occurred");
      print "error\n";
    }
   print <<ETI;
Nombre
de la carrera:     <INPUT TYPE=\"text\" NAME=\"Nomcar\" VALUE=\"$var1\"
SIZE=\"73\" MAXLENGTH=\"100\">
ETI
</SYB>
```

En los siguientes bloques, se puede observar una gran cantidad de validaciones, que se realizan para campos específicos por medio de programación en Perl:

La primera validación es cambiar / por – (diagonal por guión), se realizó con el propósito de brindarle al usuario la facilidad de proporcionar una fecha con cualquiera de los dos siguientes formatos:

Mes/Día/Año
Mes-Día-Año

Y el cambio se realiza internamente, es decir que independientemente de en que formato lo proporcione al final se suministra de la siguiente forma, lo cual se realiza con instrucciones de Perl:

```
Mes-Día-Año

$f_titula =~ s/(\/)+/\-/g;
 #print ("$f_titula \n");
```

En el caso en el que un campo si pueda ser nulo, como la fecha de titulación se suministra un valor de fecha futura con el fin de que no exista inconsistencia en la información almacenada:

```
if($f_titula eq ""){
$f_titula="01-01-2050";
}
```

Lo anterior no afecta en el sentido que el investigador pudiera en algún caso observar que existe una fecha incorrecta en sus datos, ya que el nunca la ve realmente, lo cual se puede observar en el siguiente bloque, en el que se pregunta que si una fecha es igual a *"01-01-2050" se cambie por "" (un vacío).*

```
<SYB TYPE=PERL>
$sql_stmt = qq!select convert(varchar(15),est_prof.f_titula,110) from
est_prof where cl_est_prof=convert(smallint, \"$GRADO\")!;
if (( $rc = ct_sql( $ws_db, $sql_stmt )) != CS_SUCCEED ) {
       ws_error ("Unable to process database request.");
}#Fin del if

while (($ret = ct_results($ws_db, $result_type)) == CS_SUCCEED) {
   RESULTS: {
        if ($result_type == CS_CMD_DONE) {
       last RESULTS;
        }#Fin del if
       if ($result_type == CS_CMD_FAIL) {
        last RESULTS;
        }#Fin del if
        if ( $result_type == CS_ROW_RESULT ) {
         while (@row = ct_fetch ( $ws_db )) {
         $var=@row[0];

if($var eq "01-01-2050"){
           $var="";
}#Fin del if
        }#Fin while
        last RESULTS;
       }#Fin del if
            }#Fin de RESULT
}#Fin del while

if ($ret == CS_FAIL) {
ct_cancel(CS_CANCEL_ALL);
ws_error("A database connection error occurred");
print "error\n";
}#Fin del if
  print <<ETI;
Fecha de exámen: <INPUT TYPE=\"text\" NAME=\"f_titula\"   VALUE=\"$var\"
SIZE=\"10\" MAXLENGTH=10> MM-DD-AAAA
ETI
```

```
ws_sql($ws_db, "select cred_apr from est_prof where
cl_est_prof=convert(numeric, \"$GRADO\")","
Créditos aprobados:   <INPUT TYPE=\"text\" NAME=\"cred_apr\" VALUE=\"%s\"
SIZE=\"3\" MAXLENGTH=\"3\"> %%");
</SYB>
```

El siguiente bloque se realiza con el fin de separar una fecha en Año, Mes y Día. Recordando que el formato suministrado es Mes-Día-Año:

En el caso del año, se busca un patrón de la siguiente forma: /\d\d\d\d/, es decir que contenga cuatro dígitos:

```
@fecha=$f_titula =~ /\d\d\d\d/g;
  for $token(@fecha){
      $F = $token;
}
```

Para el mes: /\A\d\d/, un patrón que contenga dos dígitos y que se encuentre al principio de la cadena (lo cual se indica con la \A):

```
@mes=$f_titula =~ /\A\d\d/g; #Estoy extrayendo el mes de inicio
  for $token(@mes){
      $M = $token;
  }
```

Para el día: /\-\d\d\-/, dos dígitos que se encuentren entre dos guiones:

```
@dia=$f_titula =~ /\-\d\d\-/g;
  for $token(@dia){
      $D = $token;
      $D=~tr/\-//d;
  }
```

Se verifica si el año suministrado es bisiesto o no, en caso de que la fecha corresponda al mes de febrero. Primero se verifica si la fecha contiene un año bisiesto, en caso de ser así se pregunta si corresponde al mes de febrero y por último se valida que no tenga un día posterior al 29. Si no es bisiesto el mes de febrero tampoco debe contener un día posterior al 28:

```
$BISI="1";
if(($F % 4 eq "0" && $F % 100 ne "0")|| $F % 400 eq "0"){
      $BISI="0";
        if($M eq "02" && $D le "29"){
         }
         elsif($M eq "02" && $D gt "29"){
        print <<ETI;
```

```
<CENTER><H4>Error en fecha de examen, el mes de febrero no tiene mas de
29 días en un año bisiesto.</H4></CENTER>
ETI
        }
}
elsif($BISI eq "1" && $M eq "02" && $D gt "28"){
  print <<ETI;
  <CENTER><H4>Error en fecha de examen, el mes de febrero no tiene mas de
28 días, en un año que no es bisiesto.</H4></CENTER>
ETI
}
```

En caso de que el mes sea: enero, marzo, mayo, julio, agosto, octubre o diciembre no pueden tener un día posterior al 31:

```
if(($M eq "01" && $D gt "31")||($M eq "03" && $D gt "31")||($M eq "05" &&
$D gt "31")||($M eq "07" && $D gt "31")||($M eq "08" && $D gt "31")||($M
eq "10" && $D gt "31")||($M eq "12" && $D gt "31")){
print <<ETI;
  <CENTER><H4>Error en fecha de examen, los meses de enero, marzo, mayo,
julio, agosto, octubre y diciembre no tiene un día posterior al
31.</H4></CENTER>
ETI
}
```

Los meses de: abril, junio, septiembre y noviembre no tiene un día posterior al 30:

```
if(($M eq "04" && $D gt "30")||($M eq "06" && $D gt "30")||($M eq "09" &&
$D gt "30")||($M eq "11" && $D gt "30")){
 print <<ETI;
  <CENTER><H4>Error en fecha de examen, los meses de abril, junio,
septiembre y noviembre no tienen un día posterior a 30.</H4></CENTER>
ETI
}
```

A su vez un mes no debe ser posterior al 12:

```
if($M gt "12"){
print <<ETI;
  <CENTER><H4>Error en fecha de examen, recuerde que no existe un mes
posterior al 12.</H4></CENTER>
ETI
}
```

Y un año debe encontrarse entre 1900 y 2060:

```
if($F lt "1900"){
print <<ETI;
  <CENTER><H4>Error en fecha de examen, recuerde que la fecha debe ser
posterior a 1900.</H4></CENTER>
```

```
ETI
}
if($F gt "2060"){
print <<ETI;
  <CENTER><H4>Error en fecha de examen, recuerde que la fecha debe ser
anterior a 2060.</H4></CENTER>
ETI
}
```

Una fecha inicial no puede ser mayor que una fecha final o en este caso no pueden ser iguales, ya que se trata del año de inicio y terminación de un estudio (licenciatura, maestría o doctorado):

```
if($f_in_est gt $f_fi_est){
   print <<ETI;
   <CENTER><H4>El año de inicio no puede ser mayor que el año de
término.</H4></CENTER>
ETI
}
       elsif($f_in_est eq $f_fi_est){
       print <<ETI;
       <CENTER><H4>Los años no pueden ser iguales.</H4></CENTER>
ETI
   }
```

El formato de la fecha debe ser de la siguiente forma: comenzar con dos dígitos seguidos de un guión posteriormente dos dígitos mas seguidos por un guión y terminar con cuatro dígitos de tal forma que se respete (Mes-Día-Año):

```
if($f_titula !~ /\d\d(\-)\d\d(\-)\d\d\d\d/){
   print <<ETI;
   <CENTER><H4>No se est&aacute; respetando el formato en fecha
inicial.</H4></CENTER>
ETI
}
```

En los campos de páginas se checa que únicamente se suministren números:

```
if($pag_ini !~ /\b[0-9]+\b/){
  print <<ETI;
  <CENTER><H4>Error en de la p&aacute;gina, se deben suministrar
únicamente n&uacute;meros.</H4></CENTER>
ETI
}
```

Se valida que no se suministren números de páginas negativos:

```
if($pag_ini lt "0"){
print <<ETI;
```

```
 <CENTER><H4>Error en p&aacute;gina inicial, recuerde que no
ex&iacute;ste una p&aacute;gina inicial negativa.</H4></CENTER>
ETI
}
```

Se eliminan todos los espacios en blanco que existan al principio de una cadena:

```
$f_titula =~ s/(\s)+//g;
```

En caso de que el usuario no proporcionara información en un campo que no fuera nulo, se valida para que el usuario se vea obligado a proporcionar la información:

```
if($Nomcar eq ""){
  print <<ETI;
  <CENTER><H4>Error en Nombre de carrera, no se admiten campos
vacíos.</H4></CENTER>
ETI
}
```

En el siguiente capítulo se presentan a manera de conclusión los comentarios finales referentes al análisis, diseño y desarrollo del sitema aquí presentado.

COMENTARIOS FINALES

Los resultados obtenidos en el sistema de la *Base de Datos Académica del Instituto de Ingeniería (BDAII)* fueron satisfactorios, ya que se logró alcanzar el objetivo planteado. Ahora los usuarios del Instituto de Ingeniería pueden consultar y actualizar su información curricular desde cualquier parte donde haya acceso a Internet, utilizando cualquier browser (Internet Explorer, Netscape Navigator, etc.), además de que el número de personas que se puedan conectar en cierto momento dado es ilimitado.

Por otra parte se ha eliminado el proceso de instalación del cliente evitándose pérdidas de tiempo y recursos. La información se captura y difunde por medio del Web y se encuentra disponible tanto para el investigador como para los directivos.

Con la migración del sistema al Internet, el Instituto de Ingeniería se coloca a la vanguardia del desarrollo de nuevas aplicaciones, teniendo en cuenta que los sistemas de información tienden día a día a ser una Intranet. Al desplegar documentos dentro de su Intranet corporativa, los usuarios podrán consultar y extraer rápidamente la información necesitada. Todos los documentos desplegados son consultados a través de un navegador de Web, dando a los generadores y usuarios de la información un medio común de comunicación corporativa.

APÉNDICE

A) Arquitectura cliente/servidor

Se trabajará en la arquitectura cliente/servidor en la cual el término cliente/servidor se refiere a la relación entre 2 procesos que cooperan trabajando juntos para resolver los requerimientos computacionales de un usuario. Expresado de una manera muy sencilla, el cliente solicita que cierta función se lleve a cabo. El servidor realiza la función. Denominada servicio. La tecnología Cliente/Servidor comúnmente es aplicada en la administración de volúmenes grandes de información. Proporciona seguridad e integridad en los datos y permite que el acceso al mismo sea mucho más rápido; permite aprovechar los beneficios que ofrecen los grandes servidores de datos en una aplicación menos costosa y obteniendo los mismos resultados, integridad, seguridad y rapidez.

La arquitectura Cliente/Servidor, es la organización de un sistema de cómputo en tres partes:

- Servidor.
- Cliente.
- Red.

A.1) Componentes de una aplicación Cliente/Servidor

En una aplicación Cliente/Servidor, los tres componentes de esta son: la interfaz al usuario (presentación), el procesamiento y el acceso a los datos. Estos componentes pueden estar distribuidos a lo largo de toda una red. La visión de la computación Cliente/Servidor se orienta hacia el soporte de ambientes complejos con múltiples plataformas, protocolos y redes, pero reteniendo la simplicidad de un sistema individual. Dependiendo de cuales funciones se encuentran en el cliente y las cuales en el servidor se tienen varias opciones de implementación Cliente/Servidor: presentación distribuida, presentación remota, funcionalidad distribuida, acceso remoto a los datos y bases de datos distribuidas. Como ya se dijo los componentes de la aplicación Cliente/Servidor son tres: administración de datos, procesamiento y presentación. Cada uno de ellos pueden encontrarse en el cliente, en el servidor o en ambas. Aunque aquí se presentan como arquitecturas "puras", pueden existir combinaciones:

- **Presentación Distribuida:** En la Presentación Distribuida, en el servidor se encuentra la administración de los datos, el procesamiento y parte del trabajo de presentación. En el cliente, se desarrolla la otra parte de la presentación.
- **Presentación Remota:** En la Presentación Remota, en el servidor se encuentra la administración de los datos y el procesamiento. La presentación se desarrolla completamente en el cliente.
- **Funcionalidad Distribuida**: En la funcionalidad Distribuida, en el servidor se encuentra la administración de los datos. Parte del procesamiento se lleva acabo en el cliente y parte en el servidor. La presentación se lleva a cabo en el cliente.
- **Acceso Remoto a los Datos:** En este caso, solo la administración de los datos se lleva a cabo en el servidor. El procesamiento y la presentación se desarrollan en el cliente.

- **Base de Datos Distribuidas:** La administración de los datos esta distribuida entre el cliente y el servidor. El procesamiento y la presentación se llevan a cabo en el cliente.

A.2) Servidor Web

Atendiendo al plano en el que nos movemos podemos tener diferentes nociones de lo que es un servidor Web: Es el conjunto de información, servicios y herramientas que definen la presencia de la empresa en Internet, es un medio de comunicación con sus clientes y prospectos, también puede convertirse en un canal comercial a través del cual proporcionar productos y servicios, es importante que en la construcción del servidor se tengan en cuenta una serie de consideraciones; ha de responder a unos objetivos y planteamientos de la empresa, debe considerar a los clientes, los contenidos que necesitan y los servicios que se les pueden prestar a través de este medio.

Sus principales funciones son las de almacenar y mantener los datos, controlar la seguridad y realizar los procesos que le sean ordenados.

A.3) Cliente Web

Es un programa con el que el usuario interacciona para solicitar a un servidor Web el envío de páginas de información. Estas páginas se transfieren mediante el protocolo HTTP (*HyperText Transfer Protocol*). Las páginas que se reciben son documentos de texto codificados en lenguaje HTML (*HyperText Markup Language*). El cliente Web debe interpretar estos documentos para mostrárselos al usuario en el formato adecuado. Además, cuando lo que se recibe no es un documento de texto, sino un objeto multimedia (vídeo, sonido,...) no reconocido por el cliente Web, éste debe activar una aplicación externa capaz de gestionarlo.

Entre los clientes Web (también conocidos como navegadores) más usuales están los siguientes:

- Netscape Navigator
- Microsoft Internet Explorer
- NCSA Mosaic
- Lynx (sólo texto)

La mayoría de los clientes soporta otros protocolos tales como: FTP, SMTP/POP, NNTP, etc.

Para desarrollar aplicaciones para el Web, debemos entender como se efectúa la interacción entre los navegadores (clientes Web) y los servidores HTTP (servidores Web). Esta interacción que envuelve tanto el camino de ida como de retorno se hace usando un conjunto de especificaciones determinadas en el protocolo HTTP; las especificaciones nos indican la información que el cliente le mandará al servidor al pedirle determinado servicio

y además la manera en que el servidor va a usar esta información para atenderlo.

En ciertos casos el pedido del cliente involucrará un determinado proceso a ejecutarse en él Servidor, como por ejemplo el acceso a una base de datos. En estos casos no es el servidor HTTP que realiza la tarea, sino un programa externo (programas de gateway) que serán invocado por él; como es costumbre el programa externo necesitará una serie de informaciones para ejecutarse de manera correcta, por lo que habrá que definir especificaciones adicionales que determinen que forma el servidor HTTP va a comunicarse con el programa externo; a estas especificaciones se les conoce como CGI (Common Gateway Interface).

Su principal tarea es la de atender las peticiones del usuario.

El cliente procesa peticiones del usuario para el servidor, cuando este las ejecuta son retornadas y desplegadas a través del cliente. Un cliente puede enviar una petición al servidor y esperar cualquier respuesta, la cual puede ser el resultado de la información o un mensaje de error.

A.4) Red

Los servicios de la red son críticos en Cliente/Servidor, casi siempre el servicio es proporcionado en Ethernet. Usando TCP/IP como protocolo, aunque no es raro el empleo de otros. La red de cómputo es el hardware y software de comunicaciones que enlaza a los clientes con los servidores.

Una buena aplicación Cliente/Servidor deberá balancear adecuadamente el uso de los tres elementos, ya que cualquier falla de diseño o implementación podrán dar al traste rápidamente con el sistema.

Dependiendo de su cobertura, las redes se clasifican en redes de área local (LAN), redes de área metropolitana (MAN) y redes de área amplia (WAN). Difieren principalmente en la distancia que pueden cubrir, la tecnología de comunicaciones que emplean, el tipo de equipo, los canales de comunicación que pueden usar y la velocidad a la que pueden operar.

Su principal tarea consiste en hacer que el servidor pueda atender muchos clientes desde el lugar en que éstos deseen conectarse y dichas conexiones pueden ser concurrentes.

La capa de interface de red es responsable de controlar el hardware de red, realizando el mapeo de direcciones IP o direcciones de hardware. Adicionalmente, la capa de red encapsula y transmite salidas de datos dentro de paquetes estandarizados y esto acepta y demultiplexa entradas de paquetes de datos.

En la Figura A.1 se muestra un esquema de la arquitectura Cliente/Servidor, en la cual, varios clientes acceden a un único servidor. Esta es la configuración usual de una pequeña red de área local (LAN, Local Area Network).

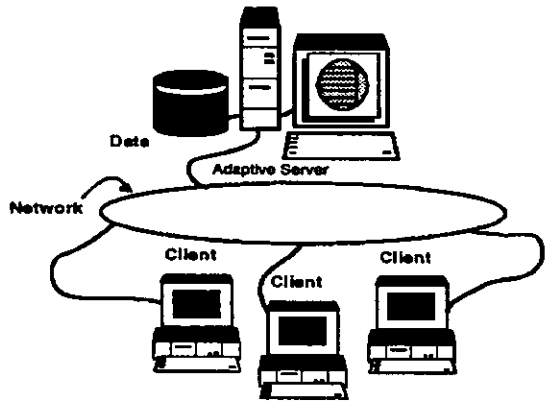


Fig. A.1 Arquitectura Cliente/Servidor

A.5) Ventajas:

Permite el acceso de un cliente a varios servidores en forma simultánea.

Permite el uso de interfaces gráficas de usuario muy versátiles y amigables en los clientes.

Opera bajo sistemas abiertos.

La carga de trabajo asociada a las aplicaciones se divide entre las distintas computadoras. Los sistemas clientes realizan parte del procesamiento, que se distribuye sobre todos los sistemas de escritorio.

A.6) Desventajas:

Las aplicaciones Cliente/Servidor suelen ser más complejas que las tradicionales en la forma anfitrión/terminales.

A.7) Beneficios Cliente/Servidor

La independencia de plataforma es probablemente una de las más grandes ventajas que tiene que ofrecer la arquitectura Cliente/Servidor.

Con series de protocolos de comunicaciones estándares, muchas aplicaciones se pueden comunicar alrededor de todo el mundo, utilizando arquitecturas de hardware completamente diferentes.

La habilidad de la tecnología del Web para plataformas independientes es más evidente por el hecho de que virtualmente hoy en día todas las plataformas de sistemas operativos incluyen soporte para acceso al Web.

B) HTML (HiperText Markup Language)

Los home pages o sitios de Internet, que usualmente se leen usando un browser, se realizan mediante un lenguaje de marcación de texto estandarizado globalmente, llamado Hyper Text Markup Language (HTML).

La gran novedad de este lenguaje es que permite definir un documento mediante parámetros claros y universales, que pueden ser leídos desde cualquier tipo de plataforma (ejemplo: PCs, Macs, UNIX, etc.) ya que se trata de simples archivos de texto, que se pueden elaborar en cualquier editor. Así mismo (dependiendo del browser) se pueden manejar imágenes, colores, tablas y por supuesto vínculos de hipertexto, que sirven para concatenar el documento a archivos de todo tipo.

B.1) Requisitos.

De manera básica para poder crear un documento HTML es necesario:

- Un programa editor de texto. Algunos de estos editores de texto son: vi, pico, notepad, Word, etc. El documento que contendrá las etiquetas de HTML debe ser escrito en un archivo de tipo texto, guardándolo con la extensión htm o html (esto depende de la computadora, pues algunas no admiten más de tres caracteres en la extensión de archivos).
- Tener conocimiento de cómo manejar algún browser HTML (Netscape, Mosaic, Internet Explorer, Netscape, etc.).
- Entender la filosofía de WWW y el significado de URL.

HTML es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces (hyperlinks) que conducen a otros documentos o fuentes de información relacionadas y con inserciones multimedia (gráficos, sonidos). La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.), así como los diferentes efectos que se quieren dar (especificar los lenguajes del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y posteriormente la presentación final de dicho hipertexto la realiza un programa especializado (como Mosaic o Netscape).

Lo que hace al HTML un lenguaje práctico y poderoso son los browsers. Estos son programas especialmente diseñados para acceder por una conexión en Internet documentos bajo el protocolo HTTP.

El browser es capaz de interpretar el diseño de un documento codificado en HTML; es el browser el que se encarga del manejo de los componentes del sistema, al creador del documento solo le corresponde diseñar lo que quiere presentar.

En la actualidad existen muchos browsers, entre los que podemos destacar los siguientes:

- **NCSA Mosaic:** Fue el precursor de todos los browsers.
- **Microsoft Explorer:** La mayor empresa de software también tiene su browser, que viene integrado con todas las utilidades de Windows 95.
- **Netscape Communicator:** Hoy en día es la herramienta más poderosa para navegar por Internet, ya que maneja los comandos más avanzados del HTML, así como animaciones gráficas y lenguajes.

B.2) Estructura básica de un documento HTML.

Un TAG de HTML es un comando o etiqueta con ciertas características propias, que denota varios elementos de un documento escrito en lenguaje HTML.

Las etiquetas consisten en el signo de menor que (<), el nombre de la etiqueta, y finalmente un signo de mayor que (>).

Las etiquetas comúnmente se presentan en pares, para darle inicio y fin a la instrucción, donde la etiqueta final es igual a la inicial solo que contiene una diagonal (/) que es la que indica que ésta es la etiqueta de fin de la instrucción.

Un documento HTML comienza con la etiqueta `<html>`, y termina con `</html>`. Dentro del documento (entre las etiquetas de principio y fin de html), hay dos zonas bien diferenciadas: el *encabezamiento*, delimitado por `<head>` y `</head>`, que sirve para definir diversos valores válidos en todo el documento; y el *cuerpo*, delimitado por `<body>` y `</body>`, donde reside la información del documento.

La única utilidad del encabezamiento en la que se hará mayor hincapié es la directiva `<title>`, que permite especificar el título de un documento HTML. Este título no forma parte del documento en sí: no aparece, por ejemplo, al principio del documento una vez que este se presenta con un programa adecuado, sino que suele servir como título de la ventana del programa que nos la muestra. Por ejemplo, el encabezamiento de un manual seria:

<title>Manual práctico de HTML</title>

Obsérvese que el título que encabeza este texto se ha escrito con mayúsculas, para distinguirlo del título *global* del documento.

El cuerpo de un documento HTML contiene el texto que, con la presentación y los efectos que se decidan, se presentará ante el *hiperlector*. Dentro del cuerpo son aplicables todos los

efectos que se van a mencionar en el resto de este documento. Dichos efectos se especifican exclusivamente a través de directivas. Esto quiere decir que los espacios, tabulaciones y retornos de carro que se introduzcan en el fichero fuente no tienen ningún efecto a la hora de la presentación final del documento. Por ejemplo, si se escribe:

```
Estas
    palabras
forman      una
  frase.
```

Se produce exactamente lo mismo que con:

Estas palabras forman una frase.

A la hora de la verdad lo que se ve es:

Estas palabras forman una frase.

En resumen, la estructura básica de un documento HTML queda de la siguiente forma:

```
<html>
<head>
<title>Título</title>
</head>
<body>
Texto del documento, menciones a gráficos, enlaces, etc.
</body>
</html>
```

B.3) Títulos.

Mediante los títulos, en sus diferentes niveles de importancia, podemos definir el *esqueleto* del documento, su estructura básica.

```
<h1>Mucha importancia</h1>
```

Mucha importancia

```
<h2>Menos importancia</h2>
```

Menos importancia

```
<h3>Mucha menos importancia</h3>
```

Mucha menos importancia

B.4) Atributos del texto

Mediante estos atributos se determina el estilo y el tipo de letra que tendrá la presentación del documento final.

El primero en analizar es el *texto normal,* entendiendo como tal el que no tiene ninguna característica especial. Para definir un párrafo como *normal,* no es necesario poner ninguna etiqueta. Lo único que hay que tener en cuenta, como ya se ha dicho antes, es que al presentar el documento se hace caso omiso de los espacios, tabulaciones y retornos de carro que se encuentren en el texto fuente. Por ello cuando se quiera forzar un final de línea es necesario utilizar dos directivas especiales: <p> para marcar un fin de párrafo, y <br> para un único retorno de carro. La diferencia entre ambas es que la separación de líneas que provoca <p> es algo mayor que la de <br>, para que los párrafos se distingan bien entre sí. Las dos directivas mencionadas se sitúan en el punto en que queremos poner la separación. Por ejemplo:

Este será un texto normal (párrafo 1, línea 1).
El primer párrafo estará formado por 2 líneas (párrafo 1, línea 2).<p>
Este ya es el segundo párrafo (párrafo 2, línea 1).<p>

Dicho ejemplo se visualizaría de la forma siguiente:

Este será un texto normal (párrafo 1, línea 1).
El primer párrafo estará formado por 2 líneas (párrafo 1, línea 2).

Este ya es el segundo párrafo (párrafo 2, línea 1).
Por supuesto, estas dos etiquetas se puede aplicar donde se desee, no sólo en el texto normal.

El *texto preformateado* (etiqueta <pre>) se aplica cuando se desea que en la presentación final del documento se respeten los espacios y retornos de carro que se hayan colocado en el texto fuente. Además se utilizará un tipo de letra de espaciado fijo, parecido al de una máquina de escribir, más pequeño que el del texto normal. Este estilo de texto puede ser adecuado, por ejemplo, para una tabla numérica sencilla:

```
<pre>
Texto preformateado
---------------------
|  1 |  2 |  3 |  4 |
|  5 |  6 |  7 |  8 |
|  9 | 10 | 11 | 12 |
---------------------
</pre>
```

Dicho ejemplo se visualizaría de la forma siguiente:

```
Texto preformateado
---------------------
|  1 |  2 |  3 |  4 |
|  5 |  6 |  7 |  8 |
|  9 | 10 | 11 | 12 |
---------------------
```

Para hacer una cita textual dentro de nuestro documento, se puede utilizar la directiva `<blockquote>`:

<blockquote>Muchos años después, frente al pelotón de fusilamiento, el coronel Aureliano Buendía había de recordar aquella tarde remota en que su padre lo llevó a conocer el hielo.
(Gabriel García Márquez, Cien años de soledad)</blockquote>

Dicho ejemplo se visualizaría de la forma siguiente:

> Muchos años después, frente al pelotón de fusilamiento, el coronel Aureliano Buendía había de recordar aquella tarde remota en que su padre lo llevó a conocer el hielo.
> (Gabriel García Márquez, Cien años de soledad)

Las *direcciones* de correo electrónico se suelen marcar con esta directiva:

<address>Dirección: sybase@tonatiuh.iingen.unam.mx</address>

Dicho ejemplo se visualizaría de la forma siguiente:

Dirección: sybase@tonatiuh.iingen.unam.mx

Se pueden dar también los atributos más tradicionales: negrita y cursiva:

Esto en negrita y esto en cursiva

Dicho ejemplo se visualizaría de la forma siguiente:

Esto en negrita y *esto en cursiva*

Se puede utilizar un tipo de letra similar al de una máquina de escribir:

<tt>Máquina de escribir</tt>

Dicho ejemplo se visualizaría de la forma siguiente:

Máquina de escribir

Para centrar texto (o, en general, cualquier cosa: un gráfico, por ejemplo) se usa la directiva :

Verde que te quiero verde

Dicho ejemplo se visualizaría de la forma siguiente:

Verde que te quiero verde

B.5) Listas.

Las listas se definen de forma muy sencilla: se dice dónde empieza la lista, dónde empieza cada punto y dónde acaba la lista. Las etiquetas que se utilizan en cada caso, deben aparecer al principio de línea, o al menos sin texto por delante (sólo espacios o tabulaciones).

Se puede recurrir a tres tipos distintos de listas, cada una con una presentación diferente: no numeradas, numeradas y listas de definiciones (glosarios).

Las listas se pueden anidar, es decir, en el lugar donde debería ir uno de los términos de la lista se coloca una nueva lista, que por supuesto no tiene porqué ser del mismo tipo.

Esto es una lista no numerada:

Tomates
Zanahorias
Puerros

Dicho ejemplo se visualizaría de la forma siguiente:

- Tomates
- Zanahorias
- Puerros

Esto una lista numerada:

Miguel Induráin
Tony Rominger
Eugeni Berzin

Dicho ejemplo se visualizaría de la forma siguiente:

1. Miguel Induráin
2. Tony Rominger
3. Eugenio Berzin

Un glosario está formado por una serie de parejas de *términos* (marcado con <dt> al principio de línea) y *definición* (con <dd>). Por ejemplo, se podría crear un pequeño diccionario con los términos perro, gato y pescado, de la siguiente manera:

```
<dl>
<dt>Perro (<i>n. masc.</i>)
<dd>Animal de cuatro patas que ladra.
<dt>Gato (<i>n. masc.</i>)
<dd>Animal de cuatro patas que maúlla y se lleva muy mal con el perro.
<dt>Pescado (<i>n. fem.</i>)
<dd>Animal que vive en el mar y está cubierto de escamas.
</dl>
```

Dicho ejemplo se visualizaría de la forma siguiente:

Perro (*n. masc.*)

Animal de cuatro patas que ladra.

Gato (*n. masc.*)

Animal de cuatro patas que maúlla y se lleva muy mal con el perro.

Pescado (*n. fem.*)

Animal que vive en el mar y está cubierto de escamas.

B.6) Enlaces y Gráficos.

Además de los muchos estilos y capacidades de presentación que ofrece HTML para estructurar el documento en sí, se dispone de varias directivas que permiten definir relaciones entre diferentes documentos y estructurar todo un conjunto de documentos para crear una unidad lógica. La facilidad para definir este tipo de enlaces es una de las razones de la potencia y versatilidad de HTML. Por la similitud de tratamiento que tienen los enlaces y los gráficos, se mencionará también en esta sección cómo pueden incluirse estos últimos en un documento.

Los enlaces en HTML se expresan rodeando con la directiva <a> el objeto (que puede ser un fragmento de texto o un gráfico) que vaya a servir como *anclaje* para el enlace. Por ejemplo, si se marca con <a> un gráfico, cuando en el documento final se pulse con el ratón sobre dicho gráfico se *saltará* al objeto referenciado en el enlace: otro documento, un vídeo musical, o un servidor de información meteorológica.

B.6.1) ¿Qué es un URL?

Para especificar de manera uniforme el objeto al que apunta un enlace, se utiliza una forma estandarizada que se denomina URL (*Uniform Resource Locator*, es decir, *Localizador Uniforme de Recursos*). Un URL está formado de la siguiente manera: *esquema://maquina/ruta* (en realidad, la barra / puede considerarse parte de la *ruta*).

El *esquema* es un nombre que identifica el tipo de servicio que va a proporcionarse en el destino del enlace. La razón de esta aparente complicación es que el WWW pretende unificar el acceso a servicios de información que previamente eran incompatibles entre sí, como ftp, gopher o telnet. El esquema más utilizado es `http`, correspondiente al propio WWW (es decir, que cualquier referencia a un documento HTML debería comenzar con `http://`). Otros esquemas muy frecuentes son `ftp`, `telnet`, `gopher` o `wais`.

La *máquina* y la *ruta* sirven para localizar el objeto al que apunta nuestro enlace. La *máquina* es la identificación del servidor en el cual está situado el objeto al que apunta el enlace. Puede ser simplemente el nombre de un ordenador (como `www.etsit.upm.es`) o también un nombre y un puerto (por ejemplo `www.etsit.upm.es:8000`).

La *ruta* es el nombre del archivo que contiene el documento en concreto, incluyendo el nombre del subdirectorio en el que se encuentra. Los diferentes nombres que constituyan la ruta completa al archivo se deben separar con la barra / (inclinada hacia la derecha), tal y como se hace en el sistema operativo UNIX (y al revés que en MS-DOS). La razón de este convenio es precisamente que la mayor parte de los servidores de WWW que hay en Internet son ordenadores basados en UNIX, debido a la gran superioridad tecnológica de este sistema sobre MS-DOS. Esto se nota también en que por lo general los nombres de los archivos no tienen muchas limitaciones: pueden ser casi tan largos como queramos, contener varios puntos, etc. Por ejemplo, el nombre de cierto archivo situado en un servidor podría ser: `/info/documentos/ciencia/física/relatividad.html`. Se debe tener en cuenta que en UNIX las mayúsculas y las minúsculas son distintas en los nombres de los archivos: no es igual PRUEBA que prueba.

La estructuración habitual de los archivos en un servidor de WWW. Para empezar, siempre hay una *página de bienvenida* (*home page*) que podría compararse con la *portada* de un periódico o revista: si no se sabe exactamente qué es lo que se busca, o no se sabe dónde encontrarlo, la portada es lo primero que se ve. Para acceder a la *home page* de cualquier servidor de WWW, basta con escribir una barra en el lugar de la ruta (es decir, se reclama

al servidor el *directorio raíz*). Por ejemplo, para acceder a la página de bienvenida de la NASA habría que contactar con `http://www.nasa.gov/`.

El resto de la información que se puede encontrar en un servidor de WWW se distribuye a partir de ese directorio raíz en distintos subdirectorios y archivos. Un convenio muy habitual relativo al nombre de los archivos es hacer que los archivos que contengan documentos HTML terminen en `.html`.

B.6.2) Enlaces.

Con lo que ya se ha comentado, se puede abordar sin problemas el asunto que originalmente se tenía en mente: cómo se introducen enlaces en un documento HTML. Para definir un enlace es necesario marcar con la directiva `<a>` el objeto del cual va a partir dicho enlace. Dicha directiva debe incluir el parámetro `href="`*`URL`*`"` para especificar el destino del enlace. Es decir, que antes del objeto elegido se debe abrir con `<a href="`*`URL`*`">`, y después cerrar con `</a>`. Por ejemplo, si se desea que el texto *pulse aquí para visitar la NASA* conduzca a la *home page* de la NASA, se debe escribir en el texto HTML:

Pulse aquí para visitar a la NASA
Lo cual producirá el resultado:
Pulse aquí para visitar la NASA

Por lo general no es tan preocupante irse tan lejos, sino sencillamente enlazar con otro documento que se encuentra en el mismo servidor, puede incluso que en el mismo subdirectorio. En este caso no es necesario escribir el *camino completo* al destino del enlace, sino que basta con dar la mínima información imprescindible. El programa que se use para leer el documento final suele ser lo bastante listo como para deducir el resto. Es decir, que si desde cierto documento queremos enlazar con otro que se encuentra en el mismo subdirectorio, basta con poner su nombre:

```
<a href="el_otro_archivo">pulse aquí</a>.
```

O si se encuentra en otro subdirectorio del mismo servidor, es suficiente con poner:

```
<a href="/la/ruta/que/sea/archivo.html">pulse aquí</a>.
```

También pueden utilizarse rutas relativas:

```
<a href="ruta/relativa/juegos.html">juegos</a>
```

B.6.3) Gráficos.

Para incluir un gráfico en un documento HTML se utiliza la directiva `<img>`. En dicha directiva debe incluirse un parámetro `src="URL"`, con el cual se indica dónde está el archivo con el gráfico concreto que se desea para el documento. Esto pone a disposición una gran flexibilidad, ya que se puede complementar el contenido del documento tanto con gráficos que se encuentren disponibles en el servidor de WWW como con una foto situada en un servidor de la NASA, por ejemplo, sin que el lector final tenga por qué apreciar ninguna diferencia.

Existe alguna limitación respecto a los formatos gráficos que los programas lectores de HTML puede interpretar sin problemas. El formato fundamental es el GIF, que cualquier programa con capacidades gráficas debería poder mostrar directamente en el texto (Mosaic y Netscape pueden hacerlo). Si se utiliza otro formato diferente, lo más probable es que cuando un lector esté accediendo al documento, el programa no comprenda ese formato y se tenga que solicitar la *ayuda* de otro programa, con lo cual al final el gráfico no se insertará en el lugar estratégico de nuestro documento, sino que aparecerá en otra ventana diferente.

Hay un parámetro optativo de la directiva `<img>` que sirve para proponer un texto alternativo a un gráfico. Este texto aparecerá cuando se esté usando para leer el HTML un programa sin capacidades gráficas (por ejemplo Lynx, que sólo trabaja con texto). Se trata de `alt="texto"`. Conviene utilizarlo cuando los gráficos sirven como origen a *hiperenlaces*, porque si no los programas sin capacidades gráficas no pudiesen mostrar los enlaces que nosotros queremos.

Como ocurría antes con los enlaces, por lo general no es necesario escribir el URL completo, sino que basta con dar la mínima información. Por ejemplo, para colocar en este punto del documento un logotipo de la NASA que se encuentre en el mismo subdirectorio que nuestros demás documentos, en el archivo `NASA.gif`, se debe escribir:

<img src="NASA.gif" alt="NASA"><p>

Lo que se traduce en:

Como se ve, se ha especificado el texto alternativo "NASA", con lo cual una persona que no dispusiera del programa adecuado hubiera visto algo parecido a `[NASA]` en lugar del dibujo.

Podemos también incluir un dibujo que esté en otro lugar especificando un URL completo, por ejemplo:
<img src="http://naic.nasa.gov/images/NASA.gif"><p>

Y además podemos hacer que un gráfico sea un enlace, utilizando la directiva <a>. En este caso no debemos olvidar utilizar la opción `alt="texto"` para que todos los usuarios puedan seguir el enlace:

```
<a href="http://www.nasa.gov/"><img src="http://naic.nasa.gov/images/nasa-logo.gif"
alt="NASA"></a><p>
```

B.7) Caracteres especiales.

Existen también ciertas limitaciones relativas al uso de ciertos símbolos que significan algo en HTML, como el de *menor que* (<) o el signo inglés de *and* (llamado *ampersand*: &).

Se tratará primero el caso más sencillo. Existe una razón evidente que impide que se puedan escribir ciertos símbolos directamente en un texto HTML, como por ejemplo el <: dichos símbolos tienen un significado en HTML, y es necesario diferenciar claramente cuándo poseen ese significado y cuándo queremos que aparezcan literalmente en el documento final. Por ejemplo, como ya se sabe, < indica el comienzo de una directiva, y, por ello, si se quiere que aparezca en el texto como tal se tendrá que *dar un rodeo* escribiendo algo que no de lugar a confusión, en este caso `<`. Los símbolos afectados por esta limitación, y la forma de escribirlos, se detallan a continuación:

- < (*Menor que*): <
- >(*Mayor que*): >
- & (símbolo de *and*, o *ampersand*): &
- " (*comillas dobles*): "

Es decir, que para escribir <"> en el texto HTML original se debe poner `<&quot;>`.

El otro caso especial se da cuando en un texto HTML se quiere escribir una eñe, por ejemplo. Existen dos formas de hacerlo. La primera, que es a la que obliga el estándar de HTML, consisten en utilizar *entidades*, es decir, palabras como las que antes se presentaron para escribir ciertos símbolos. Las entidades comienzan siempre con el símbolo *&*, y terminan con un punto y coma (;). Entre medias va un identificador del carácter que queremos que se escriba. Las entidades necesarias en nuestro idioma son:

- á: á
- é: é
- í: í
- ó: ó
- ú: ú
- Á: Á
- É: É
- Í: Í
- Ó: Ó
- Ú: Ú
- ü: ü
- Ü: Ü
- ñ: ñ
- Ñ: Ñ
- ¿: ¿
- ¡: ¡

Como puede verse, las vocales acentuadas se identifican añadiendo el sufijo *acute* a la vocal sin acentuar (puesto que se trata de un acento agudo). Para la u con diéresis y la eñe se usan *uml* tras una u y *tilde* detrás una ene, respectivamente. La equivalencia de los signos de abrir interrogación y exclamación es algo más oscura: a falta de una denominación más evidente, se tiene que usar el valor numérico de dichos caracteres en el código estándar latin1 (ISO-8859-1). Esto se puede hacer con cualquier otro carácter del código latin1, que es el código de caracteres básico en HTML, escribiendo *&#numero;*.

C) El lenguaje JavaScript.

JavaScript inició como el lenguaje para scripts de Netscape con el nombre de LiveScript, pero a raíz de que Sun respaldara el lenguaje a finales del 1995, se convirtió en JavaScript.

Sun de Microsystems no es la única compañía que da soporte a JavaScript, más de 28 compañías entre las que se incluyen América Online, Apple Computers, Oracle, Silicon Graphics, Architext y SCO anunciaron que también respaldarían a JavaScript como el estándar para Internet.

JavaScript está relacionado estrechamente a HTML y al visualizador de Netscape, ya que se utiliza para generar aplicaciones en las páginas Web.

JavaScript se presenta como un lenguaje de desarrollo de aplicaciones cliente/servidor a través de Internet.

El programa en JavaScript tiene la particularidad de que está incluido dentro del mismo documento HTML que lo presenta al usuario y no es por ello un programa aparte. Permite crear aplicaciones similares a los CGIs.

El programa en JavaScript reconoce eventos creados por el usuario, definiendo así un sistema interactivo.
Se puede por ello crear formularios que verifiquen la validez de la información e interpreten esta en el mismo programa incluido en el documento HTML sin necesidad de comunicación por la red. También se permite por medio de un código JavaScript realizar acciones particulares como ejecutar un archivo de audio, ejecutar un applet.

C.1) Requisitos.

Para poder crear un programa en JavaScript es necesario:

- Un programa editor de texto, editor de Ms-Dos, Notepad, etc.
- Tener conocimiento de HTML (Hiper Text MarkUp Language).
- Un visualizador (browser) de páginas HTML como Netscape versiones 2.0 o posteriores.

C.2) Ventajas y desventajas.

C.2.1) Ventajas.

- DESARROLLO RÁPIDO: JavaScript no requiere una compilación consumidora de tiempo, los scripts pueden desarrollarse en un periodo relativamente corto.

- FÁCIL DE APRENDER: Al aprender solo unos cuantos comandos y reglas de sintaxis sencillas es posible empezar a crear programas de gran variedad.

- INDEPENDENCIA DE PLATAFORMA: Los programas JavaScript no están atados a ninguna plataforma de hardware específica o algún sistema operativo.

- GASTOS MÍNIMOS: Los programas de JavaScript tienden a ser compactos y pequeños, lo que minimiza los requerimientos de almacenamiento y el tiempo de transferencia para el usuario.

C.2.2) Desventajas.

- RANGO LIMITADO DE MÉTODOS INTEGRADOS: El número de métodos integrados es pequeño, pero aun así permiten la interactividad con el usuario de manera apropiada.

- NO CUENTA CON OCULTACIÓN DE CÓDIGO: Como JavaScript debe incluirse como parte del código fuente HTML para un documento, no hay manera de proteger el código contra copias y de que sea reutilizado por personas que visiten las páginas.

- NO PERMITE LA IMPRESIÓN DE SU SALIDA: Esta es una gran desventaja puesto que cualquier salida que haya sido generada con JavaScript no se puede imprimir.

C.3) Conceptos básicos.

C.3.1) Javascript en HTML.

El código de JavaScript se escribe directamente dentro del archivo HTML y este debe encontrarse dentro de una etiqueta script.

C.3.2) Etiqueta SCRIPT.

La parte del archivo HTML, donde es encuentra el código en JavaScript es la siguiente:

<script language="JavaScript">
</script>

La etiqueta script tiene como atributos:

Language: Indica el lenguaje usado en el script, debido a la gran variedad que existe ahora con los lenguajes de script.

Language = "JavaScript"

El código de JavaScript es interpretado como tal cuando se encuentra entre los tags de *<script>* y *</script>*. La aparición de varios lenguajes de creación de scripts ha hecho necesario identificar ante el navegador el lenguaje que se esta empleando. A continuación se presenta la sintaxis para usar el lenguaje JavaScript dentro de HTML:

<script language="JavaScript">
< !--
Código JavaScript
// -- >
</script>

Si la página que contiene el script se usa en un navegador que no es compatible con los lenguajes de creación de scripts, los enunciados de código JavaScript se despliegan como si fueran cualquier texto en la página, agregando basura a la pantalla. Si se usan las etiquetas de comentario (<!-- y -- >), un navegador no compatible ignora la porción de script del documento, las diagonales dobles (//) que preceden al final de comentario en HTML aseguran que la etiqueta no se confundirá con un enunciado JavaScript.

Los programas JavaScript pueden incluirse en cualquier lugar en el encabezado o cuerpo de un archivo HTML. Muchas personas tienen la costumbre de incluir la etiqueta script en el encabezado del archivo HTML, siendo este el formato preferido.

Sin embargo una etiqueta script en un archivo HTML puede estar en cualquier parte y puede haber varias de ellas en un solo archivo.

C.3.3) Comentarios.

<!-- ..... -- >

Estos comentarios aseguran que otros visualizadores Web ignoraran el script por completo y no lo desplegarán, puesto que todo lo que se encuentre entre <!-- y --> deberá ser ignorado por el visualizador estándar.

// COMENTARIO DE UNA SOLA LÍNEA

La línea que empieza con dos diagonales // es un comentario JavaScript de una sola línea, similar a los que se utilizan en C++.

/*

COMENTARIO DE LÍNEAS MÚLTIPLES

*/

JavaScript también acepta comentarios de líneas múltiples estilo C, los cuales empiezan con /* y terminan con */. Los comentarios son útiles para ayudar a las demás personas a leer los programas y para que comprendan lo que hacen los diferentes comandos y funciones.

C.3.4) Tipos de datos.

JavaScript utiliza cuatro tipos de datos (números cadenas, valores booleanos y un valor nulo) para representar toda información que el lenguaje pueda manejar.

- **Números** : Cualquier número, no hay distinción entres enteros y reales.
- **Cadenas** : Una cadena de caracteres.
- **Booleanos** : Son valores booleanos (Verdadero/Falso).
- **Null** : El valor **null**

C.3.5) Operadores.

C.3.5.1) Operadores Binarios.

```
+    // adición
-    // sustracción
*    // multiplicación
/    // división
%    // módulo aritmético
```

Los operadores binarios necesitan dos operandos. El módulo aritmético es un operador de división especial que solo da como resultado el residuo de la operación.

C.3.5.2) Operadores asignación.

```
=       // asignar
+=      // adición, concatenar
-=      // sustracción
*=      // multiplicación
/=      // división
%=      // módulo
<<=     // desplazamiento a la izquierda a nivel de bits
>>=     // desplazamiento a la derecha a nivel de bits
>>>=    // desplazamiento a la derecha con llenado de ceros
&=      // and a nivel de bits
^=      // xor a nivel de bits
|=      // or a nivel de bits
```

C.3.5.3) Operadores Unarios.

```
++   // incremento
--   // decremento
!    // complemento
-    // negación unaria
```

C.3.5.4) Operadores Lógicos.

```
&&   // y
||   // o
```

C.3.5.5) Operadores de Comparación.

```
<    // menor que
>    // mayor que
<=   // menor o igual a
>=   // mayor o igual a
==   // igual a
!=   // distinto de
```

C.3.6) Variables.

Las variables son objetos de un programa cuyo valor puede cambiar durante la ejecución del programa. El nombre de las variables en una aplicación es asignada siguiendo ciertas

reglas que hacen referencia a ellas. Un identificador o nombre de variable en JavaScript puede empezar con una letra, un guión bajo ("_"), seguido de caracteres, que también pueden ser dígitos (0-9) y letras de la "a" a la "z", en mayúsculas o minúsculas. JavaScript es sensible al uso de mayúsculas y minúsculas.

El alcance de una variable es en donde puede ser usada una variable en un script. En JavaScript hay dos tipos de variable:

- **Global** : Uso de la variable en cualquier lugar de la aplicación.
- **Local** : Uso de la variable dentro de alguna función.

Para declarar una variable en JavaScript se recurre al uso de la palabra reservada **var**. Por ejemplo:

***var** nombre variable;*

var total=0;

D) PERL

D.1) ¿Qué es Perl?

Perl (Practical Extraction and Report Language) es un lenguaje de programación surgido a inicios de los años noventas, que busca antes que nada el facilitar la elaboración de tareas comunes en sistemas tipo UNIX, donde tradicionalmente las tareas de administración y proceso de datos se realiza con herramientas muy rudimentarias y por demás hostiles al usuario o administrador. Pero que se aplican sobre grandes cantidades de información (por lo regular texto) por lo que se requiere que sean de alto rendimiento.

D .2) ¿Para qué sirve?

Perl surgió como una opción para una gran cantidad de herramientas de UNIX en las cuales basa su propia sintaxis, buscando el mínimo sacrificio de su desempeño por una máxima facilidad de programación e integración, sigue la filosofía de mantener un ambiente que sea capaz de detectar y corregir pequeñas omisiones del programador, y de proporcionarle una forma abreviada de realizar múltiples tareas. En una palabra, es una utilería que pretende facilitar el proceso de grandes volúmenes de información sin sacrificar el rendimiento.

D.3)¿Dónde puede usarse?

Las plataformas donde Perl se ha desarrollado mas son los servidores UNIX, por sus necesidades de administración y lo robusto de su manejo de memoria y de procesos (requisitos de PERL hacia el S.O.) además de la facilidad de Perl para realizar los así llamados CGIs, interfaces para comunicar recursos del servidor con un servicio de Internet particular (como podría ser WWW o gopher). En otras plataformas, PC en particular, se han desarrollado versiones que mantienen un razonable grado de funcionalidad, pero en realidad, el sistema DOS no tiene un manejo lo bastante bueno de los procesos o de la memoria para permitir a PERL dar un buen desempeño, además de que no es común ver en PC necesidades de administración de la magnitud de un servidor institucional. Sin embargo, puede practicarse la programación en PERL de PC, o incluso elaborar programas de reporte en él, sin embargo, es algo que no se ha popularizado hasta hoy.

D.4) ¿Qué fuentes de información existen?

Los libros que son ya clásicos sobre Perl, son los libros del camello y la llama de la editorial Nutshell, además, existen magníficas introducciones y manuales de referencia que se pueden obtener vía Internet.

Para buscar información:
http://www.yahoo.com/Computers_and_Internet/Languajes/Perl/

Fuente de información general y referencia de documentos:
http://pubweb.nexor.co.uk/public/perl/perl.html
http://www.khoros.unm.edu:80/staff/neilb/perl/perl5.html

Documentación:
http://www-cgi.cs.cmu.edu/cgi-bin/perl-man
http://www.perl.com/perl/info/documentation.html
ftp://ftp.cdrom.com/pub/perl/CPAN/doc/manual/html/index.html
ftp://ftp.uoknor.edu/mirrors/CPAN/doc/manual/html/index.html

D.5) Filosofía de Perl.

"Hay mas de una forma de hacerlo"
-Larry Wall, autor del lenguaje de programación Perl.

Perl no establece ninguna filosofía de programación (de hecho, no se puede decir que sea orientado a objetos, modular o estructurado aun cuando soporta directamente todos estos paradigmas), los objetivos que se tuvieron en cuenta al diseñar la sintaxis de Perl fueron la facilidad de aprendizaje y de uso y la claridad de código, las cuales, son necesarias (aunque pueden escribirse programas en Perl complejos sí así se desea).

Por si fuese poco, Perl no es ni un compilador ni un interprete, esta en un punto intermedio, cuando mandamos a ejecutar un programa en Perl, se compila el código fuente a un código intermedio en memoria, se le optimiza (como si se fuese a elaborar un programa ejecutable) pero es ejecutado por un motor, como si se tratase de un interprete. El resultado final, es que se utiliza algo que se comporta como un interprete pero que tiene un rendimiento

comparativo al de programas compilados. Sin embargo, ya existen compiladores de Perl con la versión 5.

En fin, Perl no nos forza a nada, pero como es lógico hay ciertas reglas que se recomiendan seguir para facilitar nuestro trabajo:

- **Claridad.** Los programas deben de ser entendibles por la persona que suceda en tareas de mantenimiento, de lo contrario se perjudica a nuestros compañeros de trabajo.

- **Indentación.** Una costumbre ya clásica de la programación, es indentar el código dos espacios hacia adelante al abrir cada bloque, y terminar la indentación al terminar el bloque, de modo que las llaves de apertura y cierre quedan a la vista y en la misma columna, solas en sus renglones (esto incrementa algo el numero de líneas, pero facilita sobremanera la búsqueda y corrección de los diversos bloques de control).

- **Nombres de variables y demás**. Se debe procurar dar la máxima claridad a los nombres de las variables sin hacerlos demasiado grandes, el nombre de los contadores y variables que guardan valores concernientes a un pequeño segmento de código por lo regular son de un par de letras (c1, c2, ..., cx para los contadores, s1, s2, etc. para cadenas de entrada etc.), mientras que las variables que afectan a diversos segmentos (a modo de regla, que tienen su definición en una pantalla distinta a donde se usan) tienen nombres explicativos. Además, los nombres de archivos se usan con mayúsculas (ARCHENT, ARCHSAL, etc.) y las clases tienen su primera letra mayúscula.

- **Comentarios.** Para facilitar la comprensión de un programa no hay como explicarlo, y los comentarios son el medio ideal para hacerlo, hay por lo menos tres comentarios que siempre deben incluirse en un programa: Que hace el programa, Quien lo escribió y cuando inició y terminó de escribirlo, sobretodo en el contexto de una organización, estos tres simples comentarios pueden hacer la diferencia entre desechar un programa como indescifrable o dedicarle algún tiempo para revisarlo. Además, es prudente comentar dentro del código la forma en que el programa deberá ejecutarse, parámetros, y su sintaxis, así como comentar las estructuras de control como un modo de explicar la funcionalidad al detalle y recalcar con comentarios las funciones que cumplen las variables.

- **Sencillez.** Es cómodo en ocasiones el comprimir una serie de instrucciones en una sola línea, queda al criterio decidir cuando se gana en claridad con un código mas o menos extenso, pero no debe titubearse en comentar el código que sea "comprimido".

D.6) Diferencias entre Perl 4.3 y 5.X, como elegir versión

Actualmente existen dos versiones altamente populares de Perl, la 4.3 y la 5.0X, de hecho hay diferencias importantes entre una versión y otra. Por regla general las nuevas versiones son mejores que las anteriores de modo que las opaca en todo sentido. Perl no es la excepción a esta regla, el único factor que impide una transición inmediata es que no son

100% compatibles. La versión 5 de Perl es una reescritura total que ya incluye un manejo de estructuras abstractas de datos mucho mas poderoso, incluso, soporta la orientación a objetos a su manera. De modo que las librerías, por ejemplo para creación de CGIs no funcionan de una versión a otra por lo que la migración es poco practica.

Así pues, la decisión sobre que versión a utilizar depende del trabajo que haya sido realizado con anterioridad, si ya se tiene un sistema completo o grande en Perl 4, es recomendable mantenerlo en Perl 4 por el resto de su vida útil, pero para desarrollos nuevos es mucho más recomendable iniciarlos con Perl 5 o en su caso, la versión más reciente que esté disponible. Una tercera opción (que es por mucho la más recomendable si se tienen los recursos) consiste en instalar las dos versiones de modo que convivan en el sistema.

E) Protocolo de comunicación TCP/IP

Internet esta constituida sobre una colección de redes que recorren el mundo, debido a ello utiliza lenguajes complejos para comunicarse y al estar compartiendo información, se necesita usar un interprete que traduzca o un tercer idioma para que se comprenda dicha información. El objetivo de este ensayo es explicar de una manera sencilla y de una forma muy básica lo que se entiende por **protocolo de comunicación TCP/IP** en Internet. Las redes que conectan diferentes tipos de computadoras de alguna manera, algo debe mantenerlas a todas unidas. Así es que para garantizar que los diferentes tipos de computadoras pueden trabajar juntas, los programadores crean sus programas utilizando protocolos estándar. Un protocolo es una serie de reglas que describen, como deben hacerse determinadas tareas. Todos los programas de correo de Internet seguirán este protocolo cuando preparen un mensaje para su entrega.

El trabajo de Internet con TCP/IP creado en la década de los setenta, es resultado de una investigación que se hizo para la *Agencia de Proyectos Avanzados de Investigación de la Defensa de Estados Unidos*, o **DARPA** sobre la conectividad de diferentes tipos de computadoras y redes. Debido a que se emplearon fondos públicos para desarrollar TCP/IP, los estándares son **no** propietarios, esto es, que *nadie tiene derechos exclusivos de uso*. Además TCP/IP es hardware y software independientes, de manera que cualquier tipo de computadora puede conectarse a Internet y compartir información con otras computadoras. TCP/IP es un **conjunto de** protocolos de comunicación de datos. El nombre de TCP/IP proviene de los dos protocolos mas importantes: TCP (Transmission Control Protocol, Protocolo de Control de la Transmision) e IP (Internet Protocol, Protocolo Internet). En cuanto a su desempeño, TCP/IP descompone los mensajes, documentos y archivos en pequeños *paquetes* que se mueven con rapidez por las redes de Internet hacia su destino. Cada paquete contiene de uno a 1500 caracteres, incluyendo la dirección de la computadora que envía y de la que recibe.

Estos paquetes pueden viajar en forma independiente de una computadora a otra por cualquier ruta y a cualquier velocidad. Los paquetes de Internet se mueven sobre las líneas de transmisión, que son los caminos de la red. A medida que los paquetes se mueven por la red, viajan en ruteadores, que son conmutadores del sistema y hacen los cambios de caminos. En las intersecciones de las redes, los ruteadores deciden el mejor camino o ruta que habrán de tomar. Los ruteadores poseen información que ayuda a evaluar el tráfico de la red y la distancia de la siguiente computadora, de manera que puedan hacer llegar con eficiencia la información a su destino. Si un paquete se pierde o se corrompe su trayectoria, la computadora receptora mantiene una petición de retransmisión hasta que el paquete se recibe intacto. Este método de paquetes conmutables ayuda a que los datos se muevan de manera eficiente en Internet y evita la sobresaturación en cada una de las partes del sistema.

E.1) TCP / IP

TCP/IP es un conjunto de protocolos diseñados para las redes de Area Amplia (WAN). El protocolo TCP/IP está conformado por un modelo de cuatro capas: Interface de Red, Red, Transporte y Aplicación.

- **Capa de Interface de Red.** Como base del modelo está la capa de interface de red, responsable de poner y recuperar los paquetes del medio físico.
- **Capa de Red.** La capa de red es reponsable de las funciones de direccionamiento, empaquetamiento y ruteo. Hay tres protocolos en esta capa:
  - *IP* rutea y direcciona paquetes entre los nodos y redes.
  - *ARP* obtiene las direcciones de hardware de los nodos localizados en el mismo segmento.
  - *ICMP* manda mensajes y reporta errores con respecto a la entrega de paquetes.
- **Capa de Transporte.** La capa de transporte provee la comunicación entre dos nodos, está formado por dos protocolos:
  - *TCP* es un protocolo orientado a la conexión. Establece comunicaciones confiables para aplicaciones que transfieren una gran cantidad de datos al mismo tiempo o requieran una confirmación de los datos recibidos.
  - *UDP* es un protocolo no orientado a la conexión, no garantiza que los paquetes sean entregados. Las aplicaciones de UDP transfieren pequeñas cantidades de datos a la vez y son responsables de la confiabilidad de la entrega de los paquetes.
- **Capa de Aplicación.** La capa de aplicación está en la parte superior del modelo. En esta capa las aplicaciones obtienen el acceso a la red.

Cuando una aplicación transmite datos a otro nodo, cada capa añade su propia información como un encabezado. Al ser recibido el paquete la capa remueve su encabezado correspondiente y trata el resto del paquete como datos.

E.2) Internet Protocol (IP)

IP es un protocolo no orientado a la conexión, responsable principalmente del direccionamiento y ruteo de paquetes entre nodos.

Un protocolo no orientado a la conexión significa que no se establece una sesión antes del intercambio de datos. IP no garantiza la entrega, un paquete puede perderse, entregarse fuera de secuencia, duplicarse o retrasarse.

Una confirmación no es necesaria cuando un dato es recibido. El emisor o el receptor no son informados cuando un paquete se pierde o se manda fuera de secuencia. La confirmación de los paquetes es responsabilidad del protocolo de transporte de la capa superior.

Los campos añadidos al paquete recibido de la capa de Transporte como encabezado son:

- **Dirección IP Origen.** Identifica al transmisor del datagrama mediante su dirección IP.
- **Dirección IP Destino.** Identifica el receptor del datagrama por su dirección IP.
- **Protocolo.** Identifica al paquete como TCP o UDP.
- **Cheksum**. Cálculo matemático utilizado para verificar que el paquete llegó intacto.
- **Tiempo de Vida** (Time To Live). Especifica el número de segundos que un datagrama puede permanecer en el medio antes de eliminarse. Esto evita que los paquetes permanezcan indefinidamente en la red. Los ruteadores decrementan el TTL la cantidad de segundos que el datagrama se retuvo en el ruteador, mínimo un segundo.

Si IP identifica que la dirección destino es local, transmite el paquete directamente al nodo; si no, checa en su tabla de ruteo la forma de comunicarse con el nodo remoto a través de una ruta definida. En caso de no encontrarla transmite el paquete al default gateway.

IP está definido en el RFC 791.IP en el Default Gateway

Cuando un paquete es recibido en un ruteador, el paquete es interpretado por IP. IP en el ruteador realiza lo siguiente:

- Decrementa el TTL al menos un segundo en cada ruteador, o más, si el paquete permanece en el ruteador debido a un congestionamiento en la red. Si el TTL llega a cero el paquete es eliminado.
- Calcula un nuevo cheksum.
- Obtiene la dirección física del siguiente destinatario.
- IP retransmite el paquete.
- IP devuelve el paquete.

Este proceso es repetido en cada ruteador hasta que el paquete alcanza su destino.

E.3) Fragmentación y Ensamblaje

Si un paquete es muy grande para ser transmitido en el medio físico, IP lo divide en paquetes más pequeños y manejables. Cuando los paquetes llegan a su destino final, IP los ensambla reconstruyendo el paquete original. Este proceso se conoce como Fragmentación y Ensamblaje. La fragmentación usualmente ocurre en ambientes con diferentes tipos de acceso al medio físico, por ejemplo Ethernet y Token Ring.

La fragmentación y ensamblaje, se realizan de la siguiente manera:

- Un paquete de IP es recibido en el ruteador.
- IP divide el paquete.
- Se crea un nuevo encabezado para cada nuevo fragmento incluyendo:

- *Bandera,* indica que siguen otros fragmentos. No se añade una bandera al último paquete.
- *Identificador del Fragmento,* identifica al fragmento como parte de cierto paquete.
- *Fragment Offset,* indica el orden del fragmento dentro del paquete.

Aunque los paquetes viajen a través de múltiples ruteadores se ensamblan al llegar al nodo destino y se dirigen a TCP o UDP.

E.4) Transmission Control Protocol (TCP)

TCP es un protocolo confiable y orientado a la conexión. Los datos son transmitidos en segmentos orientado a la conexión significa que una sesión se estableció antes del intercambio de datos entre los nodos.

La confiabilidad se lograr asignando un número de secuencia a cada segmento transmitido. Por medio de este mecanismo, el nodo destino sabe si han llegado todos los segmentos. Una confirmación es utilizada para verificar que los datos fueron recibidos por el otro nodo. Por cada segmento enviado, el host destino debe regresar una confirmación (ACK) en un cierto periodo de tiempo. Si un ACK no es recibido o llegó dañado, los datos son retransmitidos.
TCP utiliza "byte-stream communications", esto es, los datos son tratados como una secuencia de bytes sin límite. Todos los segmentos TCP tienen dos partes: datos y encabezado. Los siguientes campos son añadidos al encabezado TCP:

- **Puerto Emisor**
- **Puerto Destino**
- **Número Secuencial:** la secuencia de bytes transmitidos en un segmento. El número secuencial es usado para verificar que todos los bytes han sido recibidos.
- **Número de Confirmación:** el número secuencial del siguiente byte enviado al nodo receptor.
- **Cheksum:** verifica que el encabezado no está corrupto.

E.5) Puertos TCP

Un puerto TCP provee una localidad definida para la entrega de mensajes. Los números de puerto inferiores a 256 son puertos comúnmente utilizados, algunos se enlistan a continuación.

Número de Puerto	Descripción
21	FTP
23	Telnet
53	Domain Name Server (DNS)

80	HTTP
139	Servicio de sesiones Netbios

TCP esta definido en el RFC 793.

E.6) TCP Three-Way Handshake

Una sesión TCP es iniciada a través del Three-way handshake. El propósito de este proceso es:

- Sincronizar los segmentos de envió y recepción.
- Informar al otro nodo de la cantidad de datos capaz de recibir simultáneamente. (tamaño de la ventana y del segmento).
- Establecer una conexión virtual.

Los siguientes pasos muestran el proceso:

1. El nodo emisor solicita una sesión enviando un segmento con la bandera de sincronización activa.
2. El nodo receptor confirma la solicitud enviando un segmento con la configuración propuesta.
3. El nodo emisor envía un segmento confirmando la configuración de la sesión.

TCP utiliza este mismo mecanismo para finalizar la conexión. Esto garantiza que ambos nodos han terminado la transmisión y todos los datos fueron recibidos.

Como conclusión, el trabajo de IP es transportar los datos en bruto de los paquetes de un lugar a otro. El trabajo de TCP es manejar el flujo de datos y asegurarse que estos son correctos. La Internet depende de miles de redes y millones de computadoras, y TCP/IP es el pegamento que mantiene todo unido.

F) El servidor HTTP Apache.

Apache HTTP Server es mantenido por el grupo Apache y es derivado del original NCSA HTTPD 1.3. Gracias a él podemos practicar la creación y publicación de documentos HTML de la misma forma que se hace en Internet con una estabilidad y eficacia ampliamente comprobada en la gran cantidad de servidores apache actualmente en uso.

F.1) Características de Apache.

Apache es uno de los mejores servidores de Web utilizados en la red Internet desde hace mucho tiempo, únicamente le hace competencia un servidor de Microsoft, el IIS.

Es un servidor de Web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1).

- Implementa los últimos protocolos, aunque se base en el HTTP / 1.1
- Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo y con la API de programación de módulos.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para solución de los mismos.

La versión actual del apache es la 1.2.4 (1.3 ya está en beta). En la nueva versión se incluyen características como el soporte para Windows NT y Windows 95, así como la inclusión de cuatro dígitos en las fechas para evitar los problemas del año 2000.

F.2) El protocolo HTTP.

Para que se pueda poner en funcionamiento un servidor de **Web** se requiere una aplicación que se conoce como un **httpd,** el cual se encarga de atender las solicitudes de las páginas que se encuentran en el **home page.** Comúnmente se ha venido usando los

httpd elaborados en NCSA, donde se pueden encontrar versiones para diversos sistemas operativos comunes en Internet.

Luego de haber conseguido un httpd, viene el momento de configurarlo, para lo cual se requiere indicar los siguientes datos:

1. Nombre con el que se indentificará al servidor (por ejemplo: www.pucp.edu.pe)
2. Directorio a partir del cual se encontrarán almacenadas las páginas, y que por lo tanto será el directorio a ser mostrado como la raíz del home page. (por ejemplo: /usr/local/httpd/htdocs).
3. Directorio a partir del cual se encontrarán almacenados los CGIs. (por ejemplo: /usr/local/httpd/htbin o /usr/local/httpd/cgi-bin).

Con estos datos se cuenta con condiciones para poner en funcionamiento un home page, para esto lo primero que se debe hacer es crear un archivo index.html en el inicio de las páginas. Este será el archivo que se mostrará cuando los usuarios accedan al servidor por primera vez, y es la presentación al exterior.

F.2.1) Métodos para efectuar una transacción HTTP.

Hay diferentes formas para efectuar la transacción HTTP; aquí se discutirán las más importantes:

Recuperación de información del servidor:

- Método GET
- Método HEAD

Envió de información al servidor:

- Método POST
- Método GET

Método GET para la recuperación de información del servidor.

El método GET se usa más comúnmente en los pedidos de los navegadores. Es el caso en el que solo se va a solicitar información residente en el servidor Web, y no se envían datos para realizar algún proceso en el servidor.

Ejemplo:

Página actual: http://www.livronline.com/cursos/CA1704/index.html
Hipervínculo activo: Syllabus
Pedido del navegador Web:

```
GET /cursos/CA1704/syllabus.html HTTP/1.0
Accept: text/plain
...
Accept */*
If-Modified-Since : Wed, 25 Sep 1997 21:04:23 GMT
Referrer: http://www.livronline.com/cursos/CA1704/index.html
User-Agent: Mozilla/4.01
línea en blanco
```

Respuesta del servidor Web:

HTTP/1.0 200 OK
Date: Fri, 26 Sep 1997 13:12:23 GMT
Server: NCSA/1.5
MIME-Version: 1.0
Content-type: text/html
Last-modified: Thu, 03 Sep 1997 16:03:09 GMT
Content-length: 254
línea en blanco
<html>
<head>
...
</html>

Respuesta del servidor Web en el caso de que el cliente tenga la última versión:

HTTP/1.0 304 Not Modified
Date: Fri, 26 Sep 1997 13:12:23 GMT
Server: NCSA/1.5
MIME-Versión: 1.0
línea en blanco.

Método HEAD para la recuperación de información del servidor.

En ciertas ocasiones se puede necesitar solo información "acerca" de un recurso en el Web, sin preocuparse por el contenido íntegro de la página; en estos casos, principalmente usados para verificación de consistencia e indexación, es más adecuado usar el método HEAD; en este método la respuesta del servidor no incluye el contenido de la página, sino apenas información relevante sobre ella.

Pedido del cliente Web:

HEAD /cursos/CA1704/index.html User-Agent: HEAD Test Agent
From: operador@abacosoft.com

Respuesta del servidor Web:

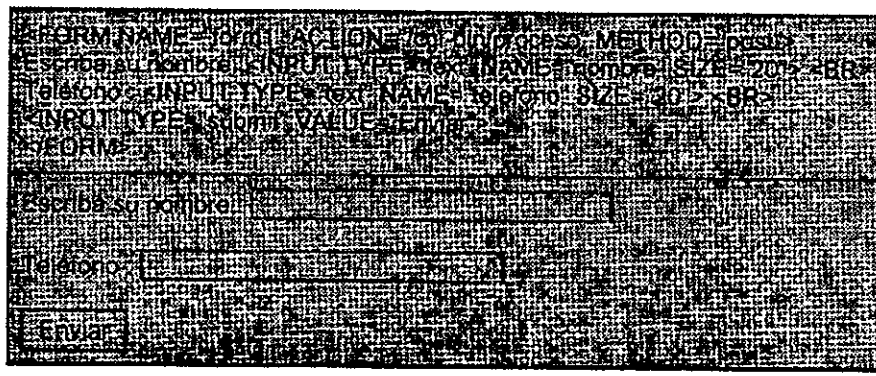
HTTP/1.0 200 OK
Date: Fri, 26 Sep 1997 13:12:23 GMT
Server: NCSA/1.5
MIME-Version: 1.0
Content-type: text/html
Content-lenght: 2331
last-modified: Fri, 15 Sep 1997 21:23:22 GMT

En casos en que la página Web tenga información adicional en el encabezado, a través del elemento <META>, esta información también será enviada por el servidor como parte de la respuesta.

Método POST para el envío de información al servidor.

Aquí está implícito que el servidor Web luego repasará la información recibida al programa externo, que se encargará de trabajar con la información que manda el cliente.

Si se cuenta con el siguiente formulario:



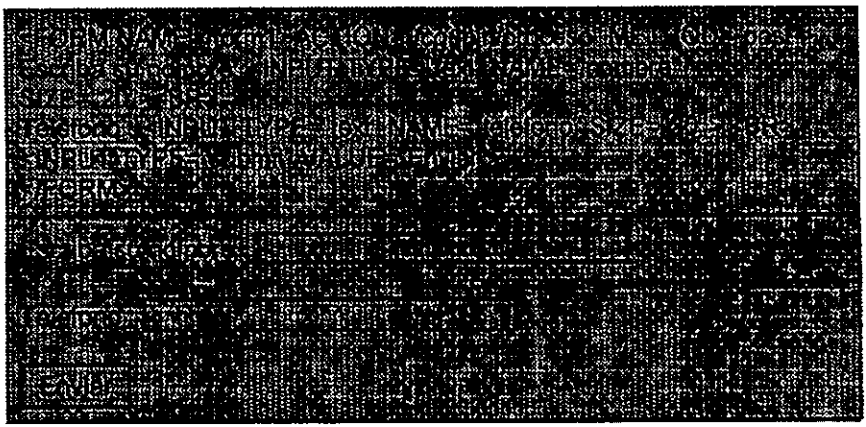
Si el usuario recibe la página Web, junto con el formulario, llenará los datos y luego presionará el botón "Enviar". El navegador enseguida enviará el pedido de servicio al servidor Web, de la siguiente forma:

POST /cgi-bin/proceso HTTP/1.0
Accept: text/plain
...
Accept */*
User-Agent: Mozilla/4.01
Content-type: application/x-www-form-urlencoded
Content-lenght: 28
línea en blanco
nombre=Jose&telefono=1234567

Método GET para el envío de información al servidor.

Las diferencias que ocurren cuando se envía el contenido del mismo formulario usando el método GET.

Repitiendo el formulario:



Usando el método GET, el pedido de servicio que envía el cliente será de la forma:

```
GET /cgi-bin/proceso?nombre=Jose&telefono=1234567 HTTP/1.0
Accept: text/plain
...
Accept */*
User-Agent: Mozilla/4.01
línea en blanco
```

Como se observa, la información del formulario es agregada al propio URL, y además el cliente no envía explícitamente el tamaño (en bytes) del contenido de los pares variable/valor.

F.3) Las preguntas más comunes sobre Apache.

1. **¿Qué es Apache?** Apache se basó originalmente en codificación e ideas basadas en el servidor HTTP más popular de todos, el NCSA httpd 1.3 (principios de 1995). Esto ha desencaminado en un sistema que puede rivalizar (y probablemente sobrepasar) a casi cualquier otro servidor basado en UNIX HTTP en cuanto a funcionalidad, eficacia y rapidez. Desde su comienzo, se ha vuelto a escribir completamente, incluye muchos rasgos nuevos.

2. **¿Por qué se creó Apache?** Apache fue creado para enviar las preocupaciones de un grupo de proveedores WWW y programadores httpd, para los cuales ese httpd no se portó como querían que se portara. Apache es producto de un esfuerzo enteramente voluntario de miembros, no por ventas comerciales.

3. **¿Cómo hace el grupo de trabajo del Apache para relatar sus esfuerzos a otros servidores tales como NCSA?** Ellos, por supuesto, tienen una gran deuda con NCSA y sus programadores, porque el servidor Apache está basado en él. De todos modos, ahora ellos tienen su propio servidor y su proyecto es mayormente de ellos. El Proyecto Apache es una empresa enteramente independiente.

4. **¿De dónde viene el nombre de Apache?** El nombre de Apache viene de "A PAtCHy server", (Un servidor lleno de remiendos). Está basado en alguna codificación existente y en una serie de archivos "parche".

5. **¿Cómo se compara Apache con otros servidores?** Apache ha mostrado ser substancialmente más rápido que muchos otros servidores libres. Aunque seguro que los servidores comerciales han exigido superar la rapidez del Apache, ellos opinan que es mejor tener un servidor libre que un servidor extremadamente rápido pero que cueste miles de dólares. Apache funciona en sitios que tienen millones de usos al día, y ejecuta sus tareas sin complicaciones.

6. **¿Cómo se ha comprobado totalmente el Apache?** Para garantizar que funciona completamente hay que tener en cuenta que Apache está en mas de 500,000 servidores en Internet, y se ha probado completamente por servidores y usuarios. Aparte, el Grupo Apache mantiene normas rigurosas antes de lanzar versiones nuevas de su servidor. Este corre sobre una tercera parte de los servidores WWW disponibles en Internet. Cuando aparecen "Bugs" el Grupo Apache lanza parches y versiones nuevas en cuanto están disponibles. La Pagina Web del Proyecto Apache (http://www.apache.org/) incluye una lista parcial de los sitios que funcionan con Apache.

7. **¿Cuáles son sus planes de futuro?**
- Continuar como un servidor público de HTTP.
- Seguir con los adelantos en protocolo HTTP y desarrollos del Web en general.
- Recoger sugerencias de los usuarios para ajustes y mejoras.
- Responder a las necesidades de un gran numero de proveedores así como a usuarios.

8. **¿A quién tengo que contactar para recibir ayuda?** No hay ningún soporte oficial de Apache. Ninguno de los desarrolladores quiere sumergirse en un mar de preguntas triviales que se pueden resolver en otra parte. Informes de "bugs" y sugerencias deben enviarse vía pagina de informe de "bugs". Otras preguntas deben dirigirse al *comp.infosystems.www.servers.unix* newsgroup donde alguien del equipo Apache que este en ese momento debería ser capaz de ayudar.

9.**¿Dónde puedo conseguir el Apache?** Puedè enterarse de como bajarse la fuente del Apache en la página del Proyecto Apache en Internet: http://www.apache.org/.

G) Sybase SQL Server.

SYBASE SQL Server system 11.5, es un sistema manejador de base de datos relacionales capaz de manejar a muchos usuarios concurrentes preservando la integridad de datos y proveyendo una administración avanzada.

Algunas de las habilidades del SQL server son las siguientes:
Protección completa de integridad de datos, para soportar transacciones complejas y seguridad avanzada para objetos que soportan sus reglas lo cual es una parte implícita de sus bases de datos.

El SQL Server 11.5, puede correr sobre sistemas operativos de: NetWare, Windows NT, OSF, Solaris, AIX y HP/UX.

G.1) Características.

Sybase soporta 2 millones de tablas por cada base de datos, 32767 bases de datos en Sybase, soporta 16 bases de datos abiertas en una misma consulta, soporta 16 tablas en una consulta a una base de datos.

Requerimientos Mínimos por Plataforma	**RAM Requerida**	**Protocolo de Red**
Sun SPARC Solaris 2.5.1	32 MB	TCP or SPX
HP 9000/700 & 800 HP UX 10.01	32 MB	TCP or SPX
IBM RS/6000 AIX version 4.1.4	32 MB	TCP or SPX
Digital UNIX 4.0A	32 MB	TCP or DECnet

Conjunto de parámetros de memoria compartida: Para que un Adaptive Server corra, el kernel del sistema operativo debe estar configurado para alojar segmentos de memoria compartida:

Plataforma	**Número de páginas de 2K**	**Cantidad de memoria**
Sun solaris	12,000	24 MB
HP/UX	9,000	18 MB
IBM RS/6000	9,000	18 MB
Digital UNIX	13,312	26 MB

G.2) Requerimiento de Hardware

- Para SQL Server 8 MB en RAM
- Espacio en disco para soportar librerías en el sistema 6 MB
- Para instalarlo completo son de 50 a 60 MB

G.3) Sybase y su uso en arquitectura Cliente/Servidor

- Diagrama Cliente/Servidor básico
- Consultas en línea
- Procesador de transacciones
- Arquitectura Cliente/Servidor
- Lenguaje: SQL, Structured Query Language
- Abierto

G.4) Productos

Los productos son los siguientes:

- Servidor de bases de datos SQL Server
- Herramientas para el uso de Sybase en terminales: DWD, APT e ISQL
- Herramientas de interoperabilidad: Open server, Omni SQL Gateway
- Herramientas de replicación: Replication Server
- Herramientas para desarrollo de aplicaciones: DB library
- Otras herramientas: BCP

H) Transact-SQL.

H.1) Introducción.

SQL (Structured Query Language - Lenguaje estructurado de consultas) es un lenguaje de alto nivel para sistemas de bases de datos relacionales. Desarrollado originalmente por el laboratorio de investigación de IBM en San José a finales de los años70, SQL ha sido adoptado y adaptado en muchos sistemas de administración de bases de datos relacionales.

Ha sido aprobado como norma oficial para lenguajes de consultas relacionales por parte del American National Standards Institute (ANSI) y la International Organización for Standarization (ISO). Transact-SQL es compatible con IBM SQL y con la mayoría de las demás implementaciones comerciales de SQL, y también proporciona importantes capacidades y funciones adicionales.

Aunque la "Q" de SQL significa "Query" (consulta), SQL incluye comandos no solo para la consulta (recuperación de datos) de una base de datos, sino también para la creación de bases de datos y objetos de bases de datos, adición de datos nuevos, modificación de datos existentes y otras funciones.

H.2) Estructura básica.

La estructura básica de una expresión en SQL consta de las siguientes cláusulas:

SELECT: El comando select sirve para consultar la información de la base de datos. Puede utilizarse para recuperar un subconjunto de filas de una o varias tablas y un subconjunto de columnas de una o varias tablas.

FROM: La cláusula from es necesaria en todas las instrucciones select que usen datos procedentes de tablas o vistas. Se utiliza para mostrar todas las tablas y vistas que contengan las columnas incluidas en la lista de selección y en la cláusula where. Si la cláusula from incluye varias tablas o vistas, se separan con comas.

El número máximo de tablas y vistas permitidas en una consulta es 16. Este total incluye las tablas indicadas en la cláusula from, las tablas de base a las que hace referencia una definición de vista, todas las tablas a las que se hace referencia en las subconsultas y todas las tablas a las que se hace referencia como parte de las restricciones de integridad referenciales.

WHERE: La cláusula where de una instrucción select especifica los criterios que definen las filas exactas que deben recuperarse.

H.3) Operadores de comparación de SQL.

Operador:	**Significado:**
=	Igual que

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
!=	Distinto de

H.4) Operadores lógicos.

Los operadores lógicos **and, or** y **not** se utilizan para conectar condiciones de búsqueda en las cláusulas where.

And: Combina dos o más condiciones y devuelve resultados solo cuando todas las condiciones se cumplen.

Or: Conecta dos o mas condiciones, pero devuelve resultados cuando se cumple cualquiera de las condiciones.

Not: Niega la expresión a la que precede.

Los operadores aritméticos y basados en bits se evalúan antes que los lógicos. Cuando se usa mas de un operador lógico en una instrucción, **not** se evalúa primero, luego **and** y por último **or**.

H.5) Comandos de SQL estándar.

Estos comandos son usados para crear, almacenar, cambiar, recuperar y dar mantenimiento a la información en una base de datos relacional.

alter: Altera la estructura de una tabla.
audit: Activa el proceso de auditoría interna.
commit: Realiza físicamente el cambio a la base de datos, pero para evitar problemas se debe salvar la información.
comment: Para escribir documentación en los programas.
create: Crea los objetos de la base de datos.
delete: Elimina registros de una base de datos.
drop: Elimina objetos de una base de datos.
grant: Asigna privilegios de un usuario sobre un objeto.
insert: Inserta datos dentro de una tabla.
lock: Coloca un candado en un objeto.
no audit: Desactiva el sistema de auditoria de la base.
rename: Renombra a un objeto de la base de datos.
revoke: Elimina privilegios a un usuario sobre un objeto.
rollback: Permite regresar al estado anterior de la base de datos, es decir hasta el **commit** anterior.
select: Ejecuta una consulta a la base de datos.

update: Permite hacer modificaciones a los registros o datos de una tabla.
validate: Válida la información de la base de datos con respecto a un dominio especificado.

H.6) Tipos de datos del sistema SQL server.

Tipos de datos:	Sinónimos:	Margen:	Bytes de almacenamiento:

H.6.1) Números exactos: enteros.

Tinyint	Integer	0 a 255	1
Smallint	Integer	2^{15} -1(32.767) a -2^{15} (-32.768)	2
int	Integer	2^{31} – 1(2.147.483.647) a – 2^{31} (-2.147.483.648)	4

H.6.2) Numéricos exactos: decimales.

Numeric (p, s)	Decimal	10^{38} –1 a -10^{38}	2 a 17
Decimal (p, s)	Decimal	10^{38} –1 a -10^{38}	2 a 17

H.6.3) Numéricos aproximados.

float (precisión)		dependiente de la máquina	4 u 8
Double precision		dependiente de la máquina	8
Real		dependiente de la máquina	4

H.6.4) Monetarios.

Smallmoney		214.7483647 a – 214.748,3648	4
Money		922.337.203.685.477, 5807 a – 922.337.203.685.477, 5808	8

H.6.5) Fecha/hora.

Smalldatetime		Del 1 de enero de 1900 al 6 de junio de 2079	4
Datetime		Del 1 de enero de 1753 al 31 de diciembre de 9999	8

H.6.6) Caracteres.

char(n)	Character	255 caracteres o menos	n
Varchar(n)	Character varying, char varying	255 caracteres o menos	longitud de entrada real
nchar(n)	National character, national char	255 caracteres o menos	n * @@ncharsize
Nvarchar(n)	Nchar variyng, national char	255 caracteres o menos	@@ncharsize * número de caracteres
Text	Varying, national character varying	2^{31} – 1(2.147.483.647) bytes o menos	0 o múltiplo de 2K

H.6.7) Binarios.

binary(n)		255 bytes o menos	n
Varbinary(n)		255 bytes o menos	longitud de entrada real
Image		2^{31} – 1(2.147.483.647) bytes o menos	0 o múltiplo de 2K

H.6.8) Bit.

Bit		0 ó 1	1 (un byte contiene hasta 8 columnas bit)

I) Web.sql

I.1) ¿Qué es Web.sql?

Sybase Web.sql es una herramienta para Internet e Intranets con una nueva tecnología que facilita el acceso a bases de datos relacionales desde el World Wide Web, así como también la creación dinámica de documentos personalizados en formato HTML. El resultado es un mejor rendimiento en el acceso a la base de datos y en el tiempo de respuesta. Además, Web.sql integra la tecnología Open Client de Sybase, lo cual permite que los datos de cualquier fuente sean incorporados dinámicamente en las páginas Web. Con Web.sql se extienden las funciones de un servidor Web posibilitando la inserción de instrucciones de base de datos, tales como sentencias SQL, así como también código de escritos Perl, dentro de un documento en formato HTML.

I.2) Características y Funciones.

La interfaz Web.sql reconoce dos tipos de extensiones de archivos: .hts (Hyper Text Sybase) y .pl (Perl).

Cuando los archivos tienen la extensión .hts, el cliente browser recibe los resultados en formato HTML puro, sin embargo si se requiere que el cliente browser reciba otro tipo de documentos, tales como los archivos con extensiones .gif, entonces se debe especificar el tipo de contenido de dicho documento en un archivo Perl (.pl), antes de enviar los datos al browser.

Además, Web.sql posibilita la ejecución de un escrito Perl, el cual a su vez puede devolver texto, gráficos, sonido, vídeo o cualquier otro tipo de dato que el Web browser soporte o pueda transferir a una aplicación auxiliar.

Un escrito Perl tiene la facultad de interactuar con una base de datos, tal como lo hace un archivo HTS a través de sentencias SQL. Por ejemplo se puede emplear un escrito Perl para extraer una imagen de una base de datos.

Esto se logra a través de las dos interfaces de programación de aplicaciones API que vienen con Web.sql: Convenience API y Client-Library API. Dichas interfaces proveen un conjunto de rutinas o funciones para ejecutar las operaciones más comunes en un archivo HTS, así como también para llevar a cabo la mayor parte de la interacción con una base de datos. Adicionalmente, estas interfaces posibilitan manejar los datos de los registros que devuelve el servidor SQL.

I.3) ¿Cómo funciona Web.sql?.

La Figura 1, muestra la arquitectura típica de un servidor Web, donde el Web browser hace una requisición de documentos en dicho servidor especificando un URL.

El servidor HTTP interpreta este URL en una ruta para un archivo específico que se encuentra en la máquina servidora que es anfitriona. Si el archivo contiene las extensiones .html, .gif, .jpeg o de cualquier otro tipo que el servidor Web reconozca, el servidor HTTP devuelve el archivo requerido directamente al browser.

En cambio, si el archivo requerido es un programa que reside en un directorio autorizado, el servidor HTTP ejecuta dicho programa de acuerdo al CGI y envía los resultados del programa al Web browser.

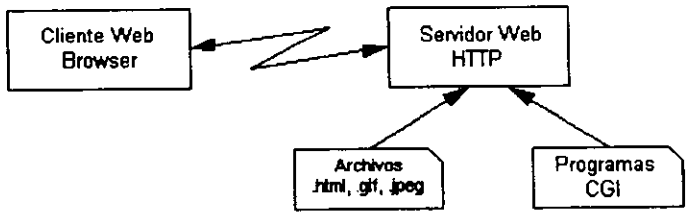


Figura 1: arquitectura típica de un servidor Web.

La Figura 2, muestra la arquitectura de un servidor Web con la interfaz Web.sql, donde dicho servidor maneja las requisiciones de archivos HTML, tal como se indicó en la Figura 1. Sin embargo, cuando un cliente browser hace una requisición del URL, que es interpretado como un archivo con extensión .hts o .pl, el servidor HTTP pasa la requisición al programa Web.sql. Éste lee el archivo especificado, luego procesa las requisiciones a la base de datos y del código Perl contenidas en el archivo. Finalmente compone los resultados en formato HTML para que el servidor HTTP los pase al browser.

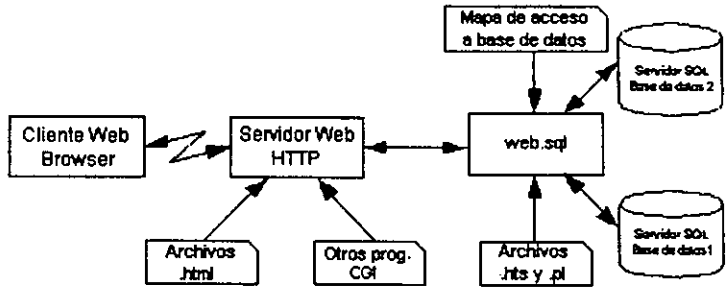


Figura 2: Arquitectura de un servidor Web con Sybase Web.sql.

I.4) Formato de archivos Hyper Text Sybase (HTS).

El formato de archivos HTS es una extensión del formato estándar HTML. Se puede incluir cualquier cosa en un archivo HTS inclusive extensiones de HTML, tales como viñetas Java o de JavaScript. El programa Web.sql ignora estas viñetas y las remite al servidor HTTP.

Para la funcionalidad de Web.sql, los archivos HTS soportan todas las viñetas de HTML 3.0 y una adicional: <SYB>. Este programa interpreta todas las líneas entre las viñetas <SYB> y </SYB>, ya sean éstas sentencias de instrucciones SQL o código Perl. Luego inserta los resultados de esas sentencias en el flujo del HTML que envía al servidor HTTP.

La viñeta <SYB> posee un atributo opcional denominado TYPE. Al incluir "TYPE=SQL", se indica que el bloque <SYB> contiene únicamente sentencias SQL. De igual forma "TYPE=PERL" indica que el bloque <SYB> contiene exclusivamente código Perl. Por cierto, las sentencias Perl pueden contener llamadas a funciones que ejecutan sentencias SQL.

Si no se incluye el atributo "TYPE", el procesador Web.sql asume que el bloque <SYB> contiene sentencias Perl.

El tipo de contenido que devuelve el formato de archivo HTS para los resultados es exclusivamente HTML. Sin embargo, si es necesario devolver algún otro tipo de contenido, se requerirá de un archivo Perl (.pl) donde se especifique el tipo de contenido, haciendo uso de la función ws_content_type().

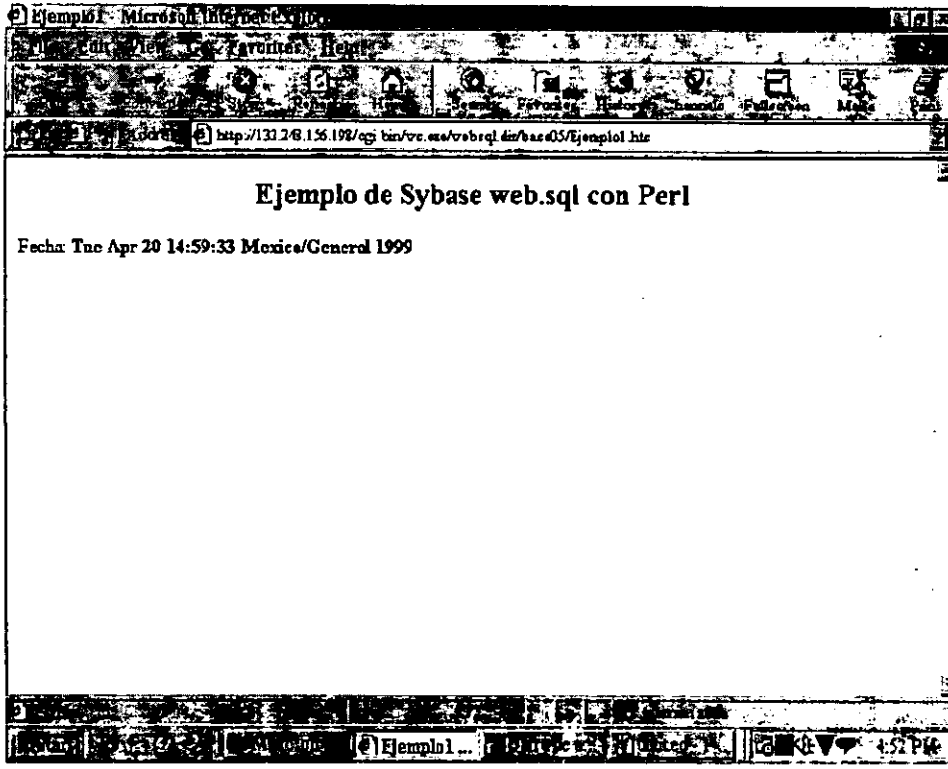
Para el caso, dentro del archivo Perl grafico.pl, se especifica el contenido image/gif para el archivo foto.gif antes de imprimirlo.

```
Ws_content_type("image/gif");
Print 'foto.gif';
```

En el siguiente ejemplo, se obtiene la fecha y la hora del sistema en el servidor y se imprime el resultado en salida estándar, acompañado de algunas viñetas de HTML:

```
<HTML>
<HEAD><TITLE>Ejemplo</TITLE></HEAD>
<BODY>
<H2><CENTER>Ejemplo de Sybase  Web.sql con Perl.</CENTER></H2>
<SYB TYPE=PERL>
require "ctime.pl";
print "<P>";
print "Fecha: ", "<STRONG>", &ctime(time), "</STRONG>";
print "<P>";
</SYB>
</BODY>
</HTML>
```

El archivo HTML resultante, enviado al browser, es el siguiente:



Otro ejemplo de Archivos HTS, pero con bloques de SQL, seria el siguiente:

```
<HTML>
<HEAD><TITLE>Ejemplo de SQL en archivos HTS</TITLE></HEAD>
<BODY>
<SYB TYPE=SQL>
select * from ACTOR
</SYB>
</BODY>
</HTML>
```

Nos da como resultado toda la información que existe en la tabla ACTOR.

I.5) Acceso a los archivos HTS.

Los archivos HTS se abren en un Web browser a través de un URL. Por ejemplo, para abrir la página de bienvenida de Sybase Web.sql, se utilizan los siguientes URLs, dependiendo de la plataforma, en éste caso para Unix:

- Versión NSAPI:

http://<servidor-www>/<dir-websql>/welcome.hts

- Versión CGI:

http://<servidor-www>/<dir-cgi>/websql/<dir-websql>/welcome.hts

Donde:

<servidor-www>: es el nombre del directorio de escritos CGI del servidor Web.
:<dir-cgi>:
ws.exe ó Websql: es el nombre del programa CGI de Web.sql.
<dir-websql>: es la ruta del subdirectorio de Websql en el directorio raíz de los documentos HTML del servidor Web.
Welcome.hts: es el nombre de la página de bienvenida de Sybase Web.sql.

I.6) Compatibilidad de Sybase Web.sql.

La herramienta Web.sql se integra perfectamente con capacidades propias para operar con los siguientes productos Sybase:

- Sybase IQ: que posee funciones sofisticadas para el análisis de datos.
- Replication Server: un motor para la réplica de base de datos en tiempo real.
- SQL Server: Soportado masivamente en sistemas paralelos.
- CONNECT: que provee acceso a múltiples fuentes de datos.

Sybase Web.sql funciona bajo las siguientes plataformas o sistemas operativos:

- Sun Solaris (SPARC) de Sun Microsystem.
- Windows NT de Microsoft.
- IRIX de Silicon Graphics.
- HP-UX de Hewlett Packard.

I.7) Requerimientos.

Web.sql 1.1 para Sun Solaris:

- Hardware: Sun SPARC estación.
- Sistema operativo: Solaris ver. 2 .4, 2.5 y 2.5.1
- Memoria: 16 MB (mínimo).
- Espacio libre en disco: 14 MB.
- Base de datos: Sybase SQL Server ver. 10 en adelante.
- Sybase IQ.

Servidor Web:

- Para la versión CGI: Cualquier servidor Web que soporte CGI ver 1.1 (Netscape, NCSA, etc.).
- Para la versión NSAPI: Netscape Communicator Server ver. 1.10, Commerce Server ver. 1.12, Enterprise Server ver. 2.0.

I.8) Interfaces de Programación de Aplicaciones (API).

Web.sql provee dos Interfaces de Programación de Aplicaciones (API) en Perl para la interacción con bases de datos, las cuales son: "Convenience" API ("conveniente") y Client-Library API (bibliotecas para cliente). Si no se ha trabajado antes con Sybase Open Client, es mejor utilizar la interfaz "Convenience" API, puesto que requiere de menos programación.

En la interfaz "Convenience" API, todas sus funciones contienen el prefijo "ws_". En cambio, la interfaz Client-Library API se puede utilizar para manejar los datos devueltos por el servidor SQL, registro por registro. Todas las funciones contienen el prefijo "ct_".

I.8.1) Uso de la Interfaz Convenience API.

ws_connect:
Conecta a un servidor SQL y devuelve un identificador de la conexión.

$handle = ws_connect([$connection_name])

Parámetros:
$connection_name: es el valor de la conexión a la base de datos definida en el mapa de acceso en web.sql.pl

Valor que regresa:
$handle: una conexión determinada para usar en otras llamadas a un API de web.sql

ws_content_type:
Fija el tipo de contenido para los datos devueltos por un archivo con extensión .pl.
ws_content_type($data_type)
Parámetros:
$data_type: una cadena especificando el formato de datos en términos de MIME content_type.

ws_error:
Imprime un mensaje de error y una cadena opcional. Termina el procesamiento de la página actual.

ws_error ($string)

Parámetros:
$string: es la cadena.

```
If(!$nombre){
    ws_error("Debe especificar un nombre");
        }
```

ws_fetch_rows:
Obtiene e imprime los registros devueltos por la función ct_sql.
ws_fetch_rows($handle[,$format])

Parámetros:
$handle: conexión valida al servidor de base de datos.
$format: un formato de salida opcional para el dato que trae, donde la sintaxis del dato es similar a la de printf.

ws_print:
Imprime una cadena, extendiendo las referencias a variables Perl.

ws_print($string)

Parámetros:
$string: es la cadena a imprimir.

ws_sql:
Ejecuta uno o más comandos SQL e imprime los resultados.

ws_sql($db, $sql, [,format])

Parámetros:
$handle: Típicamente se usa $ws_db como argumento de ($db) para especificar la base de datos para un archivo HTS.

$sql: Es una cadena que contiene uno o más comandos de sql. Si falla el comando, ws_sql imprime un mensaje de error.
$format: El formato de la cadena es opcional. Si no se incluye un formato de cadena, ws_sql le da formato a los renglones que regresa de resultado como una tabla de HTML 3.0

```
<SYB TYPE=PERL>
$sql_stmt=qq! Select cve_actor, nom_real, nom_artis, direccion from ACTOR!
$format="%s%s Nombre artístico y dirección: %s, %s.<P>\n";
print "<P><HR>";
</SYB>
```

Cada renglón aparecerá como un párrafo separado.

ws_rpc:
Ejecuta un procedimiento almacenado y registrado, actualiza los argumentos e imprime los resultados.

ws_rpc($handle, $rpc_name, @params[,$format])

Parámetros:
$handle: El argumento $handle puede ser cualquiera de las bases de datos manejadas ($ws_db) para el archivo HTS u otra base de datos que se obtenga con la rutina ws_connect o ct_connect.
$rpc: Nombre del procedimiento almacenado.
Lista de parámetros:
@params: (posición de parámetros), solo se puede pasar el valor del argumento encerrado entre corchetes []. Se pasa el argumento en la primera posición como el primer parámetro del procedimiento almacenado y así sucesivamente.

Para enviar o recibir un tipo de dato en particular:
{
'type ' => CS_<datatype>,
'value' => <scalar> o <scalar_ref>
'scale' => CS_MAX_SCALE o CS_DEF_SCALE,
'precision' => CS_MAX_PREC o CS_DEF_PREC
}

I.8.2) Uso de la Interfaz Client-Library API.

Posibilita el manejo de los datos de los registros devueltos por el servidor. Provee de una interfaz similar a la de Sybase Open Client Client-Library (CT Lib) API.

ct_callback:
Instala e invoca una rutina para el manejo de errores.
ct_callback($type, $cb func)

Parámetros:
$type: el tipo de rutina callback de interés. La siguiente tabla lista los valores simbólicos que son válidos para $type:
CS_CLIENTMSG_CB: ct_callback instala un mensaje callback en el cliente.
CS_SERVERMSG_CB: ct_callback instala un mensaje de callback en el servidor.

$cb_func: una cadena especificando el nombre de una subrutina de Perl. Es la rutina callback a instalar.

ct_cancel:
Cancela un comando o los resultados de un comando.
$rc=ct_cancel($handle, $type)

Parámetros:
$handle: variable predefinida para la conexión a la base de datos.
$type: El tipo de cancel:
CS_CANCEL_ALL: ct_cancel envía un mensaje al servidor, cancela los comandos. Client-Library inmediatamente cancela todos los resultados generados por el comando.
CS_CANCEL_ATTN: ct_cancel envía un mensaje al servidor para cancelar el comando. La siguiente vez la aplicación lee desde el servidor, generados por el comando cancelado.
CS_CANCEL_CURRENT: cancela el resultado solamente.

Valores que regresa:
CS_SUCCEED: La rutina fue completada exitosamente.
CS_FAIL: La rutina falló.

ct_col_types:
Recupera un arreglo de tipos de columnas para los resultados de la consulta actual.

%types | @types = ct_col_types($handle [,$doassoc])

Parámetros:
$handle: valor predefinido para la conexión a la base de datos.
$doassoc: una bandera que determina que rutina regresa un arreglo asociativo o un arreglo regular.

Valores para $doassoc:
1: Regresa un arreglo asociativo de tipo de columnas. Los elementos del arreglo son indexados por los nombres de columnas.
No especificado: regresa un arreglo de tipo de columnas.

Valores que regresa ct_col_types:
%types: un arreglo asociativo de un tipo de columna. Los índices del arreglo son los nombres de columnas.
@types: Un arreglo de tipos de columna.

ct_col_names:
Recupera un arreglo de nombres de columnas para los resultados de la consulta actual.
ct_connect: establece una conexión de base de datos a un servidor.

@names=ct_col_names($handle)

Parámetros:
$handle: un valor predefinido para la conexión a la base de datos.

Valores que regresa:
@names: un arreglo de nombre de columnas para el conjunto de resultados.

ct_fetch:
Obtiene un solo registro de los datos resultantes.

@row | %row = ct_fetch($handle[, $doassoc])

Parámetros:
$handle: conexión válida para el servidor de la base de datos y login.
$doassoc: una bandera que determina que recibir, un arreglo asociativo o un arreglo regular.

Valores válidos para $doassoc:
1: El renglón de datos es regresado en un arreglo asociativo.
No especificado: El renglón de datos es regresado en un arreglo regular.

Valores que regresa ct_fetch:
%row: un arreglo asociativo que contiene un renglón de datos del conjunto de resultados. Los índices del arreglo son los nombres de las columnas.
@row: Un arreglo regular que contiene un renglón de datos del conjunto de resultados.

ct_fetch_parameters:
Actualiza las variables de parámetro de salida provistas a ct_rpc.

ct_fetch_parameters($handle)

Parámetros:
$handle: conexión válida al servidor de la base de datos.

ct_options:
Fija, recupera o limpia los valores de las opciones del procesamiento de consultas del servidor.

ct_options($handle, $action, $option, $param, $type)

Parámetros:
$handle: conexión válida al servidor de la base de datos.
$action: uno de los siguientes valores simbólicos que se listan a continuación:

CS_SET: ct_options, conjunto de opciones. Result data es sustituido por $rc.
CS_GET: ct_options recibe la opción. Result data es sustituido por la variable $param.
CS_CLEAR: ct_options por un valor de default. Los valores de default son determinados por el servidor para el cual una aplicación es conectada. Result data es sustituido por $rc.

$option: la opción de interés, existen muchos parámetros para ct_options.
$param: Todas las opciones toman parámetros. Cuando se pone una opción, $param puede ser un valor entero o una cadena de caracteres. Cuando se limpia una opción $param vale 0.
$type: CS_INT_TYPE si $param es un entero; CS_CHAR_TYPE si $param es una cadena de caracteres.

La siguiente es la lista de parámetros que puede regresar ct_options:
$rc: el éxito de la llamada a ct_options. CS_SUCCEED indica que la rutina fue un éxito, CS_FAIL indica que la rutina falló.
$result: el valor de $param especificado en la rutina. El resultado es una cadena vacía si se pone el parámetro.

ct_res_info:
Recupera información acerca del conjunto de resultados o comando actuales.

$res_info=ct_res_info($handle, $info_type)

Parámetros:
$handle: Conexión válida al servidor de base de datos.
$info_type: el tipo de información que regresa. La siguiente tabla lista los valores simbólicos que son válidos para $info_type.
Valores que regresa ct_res_info:
$res_info: el regreso de la información solicitada.

ct_results:
Determina el estado del comando SQL y el tipo de resultados devueltos.

$rc=ct_results($handle, $result_type)

Parámetros:
$handle: conexión válida al servidor de la base de datos.
$result_type: una variable entera en la que ct_results indica el tipo de resultado disponible para el servidor de la base de datos. Este parámetro es actualmente un parámetro de salida cuando ct_results es llamado.

Valores que regresa ct_results:

CS_SUCCEED: el resultado de un proceso es disponible.
CS_END_RESULTS: Todos los resultados pueden ser completamente procesados.
CS_FAIL: La rutina falló. Si ct_results regresa CS_FAIL, una aplicación debe llamar ct_cancel con $type como CS_CANCEL_ALL antes usando la estructura del comando afectado para enviar otro comando.
CS_CANCELED: Los resultados son cancelados.

ct_rpc:
Llama a un procedimiento almacenado que reside en un servidor remoto.

ct_rpc($db_handle, $rpcname, @params o %params)
Parámetros:
$db_handle: conexión válida al servidor de base de datos.
$rpc_name: el nombre del procedimiento remoto a llamar.
%params o @params: la lista de parámetros para el RPC, incluyendo salida de parámetros.

ct_sql:
Envía uno o más comandos SQL al servidor de bases de datos.

$rc=ct_sql($handle, $query)

Parámetros:
$handle: conexión válida al servidor de base de datos.
$query: una o más sentencias de SQL.

Valores que regresa:
CS_SUCCEED: la rutina se completó exitosamente.
CS_FAIL: la rutina falló.
CS_CANCELED: la operación fue cancelada. Solo un tipo de CS_CANCEL_CURRENT de cancel puede ser cancelado.

GLOSARIO

Arquitectura Cliente/Servidor

Es una arquitectura para un sistema de cómputo en el cual el cliente hace peticiones de un servicio y el servidor provee ese servicio. Cada máquina puede especializarse dentro de las tareas de la mejor manera.

Base de datos

Un conjunto de relaciones entre datos, tablas y otros objetos de la base de datos que son organizados y presentados al servidor para un propósito específico.

Campo

Elemento de información contenido dentro de un renglón o registro. Equivalente lógico de una columna.

Cliente

Sistema que recibe recursos de un sistema remoto llamado servidor a través de una red.

Columna

Conjunto de todos los renglones de una tabla que tienen un atributo común. Contiene un dato individual dentro de cada renglón o registro.

Concurrencia

Son múltiples accesos por diferentes usuarios a la misma información.

Consistencia

La información es consistente, si al consultar datos en cualquier parte del sistema, estos son los mismos.

Constraint

Una regla aplicada a un objeto de la base de datos que asegura que todas las entradas de objetos en la base de datos satisfacen esa particular condición. Por ejemplo, una columna puede tener constraint que requieren que todos los valores en la columna sean únicos.

Dato

Representación codificada de información para usarla en una computadora. Los datos tienen atributos como tipo y longitud.

Default

Los defaults especifican los datos que deben de estar dentro de una columna si el usuario no especifica una entrada. Una columna puede tener un solo default.

Dependencia funcional

El atributo A de una relación es funcionalmente dependiente del atributo B, si el valor A esta determinado por el valor de B.

Dirección electrónica

Dentro de las comunicaciones electrónicas como el correo electrónico, ésta específica el nombre de la máquina destino y el usuario que recibirá un mensaje. En Internet esta dirección tiene el formato usuario@máquina_destino.

Dirección IP

Identificador universal de 32 bits asignando a una máquina que forma parte de Internet, consiste de una parte que identifica a la red a la cual pertenece a la máquina y una parte que identifica a la máquina misma.

Elemento

Sinónimo de campo de una tabla. Intersección de un renglón y una columna.

Entidad

Es la unidad básica que descrita por las tablas en una base de datos relacional.. Se identifica como el primer paso para el diseño lógico.

FTP

Una parte del protocolo de transferencia de archivo (File Transfer Protocol) que permite al usuario conectarse como individuo a un host remoto utilizando un login.

Hardware

Parte física de un sistema de cómputo.

Hipertexto

El concepto de Hipertexto se refiere a datos contenidos en un documento que están organizados para proveer ligas entre documentos, de modo que relacionan conceptos y temas que pueden ser ligados entre sí.

HTML

HyperText Markup Language, lenguaje de documentos de hipertexto.

HTTP

HyperText Transport Protocol, Protocolo de transportación de hipertexto.

Inconsistencia

Obtener diferentes salidas para repeticiones similares en un mismo momento.

Índice

Un objeto de la base de datos que consiste de un valor de clave de los datos de la tabla y apuntadores a las páginas que contienen esos valores. Los índices agilizan la velocidad de acceso a los datos por apuntadores a la localización de la columna en la tabla y a su vez del dato en el disco.

Integridad
: Que la BD diga siempre la verdad. Una BD es integra si concuerda con la realidad.

Internet
: Colección de redes conectadas con gateways y ruteadores. Colección de hosts conectados de varias redes regionales alrededor del mundo que utilizan protocolos TCP/IP.

IP
: Acrónimo de Internet Protocol, IP es uno de los principales protocolos que forman a la familia TCP/IP.

LAN Local Area Network
: Red de área local para que una red pueda ser considerada dentro de esta clasificación debe tener un área menor a 10 Km a la redonda.

Login name
: Nombre que define al usuario de un sistema, con éste y el password el usuario puede tener acceso al sistema.

Llave
: Un campo usado para identificar el record, a menudo se usa como índice del campo para una tabla.

Llave foránea
: Uno o más campos usados para identificar un registro, frecuentemente se utiliza como índice de una tabla.

Llave primaria
: Columna o combinación de columnas que identifican de manera única una tabla. Siempre deben ser diferentes de "nulo" y tener un índice único. Una llave primaria se usa para relacionarse con llaves foráneas de otras tablas.

Nulos
: No tienen valor explícito asignado. Nulo no es equivalente a cero o blanco.

Primera forma normal
: Una relación está en la primera forma normal si todos los campos en cada registro contienen un solo valor tomado de sus dominios respectivos.

Procesamiento distribuido
: Las tareas de procesamiento son efectuadas por uno o más sistemas interconectados mediante una red de cómputo.

Procedimiento almacenado

Una colección de comandos SQL y comandos opcionales de control de flujo almacenado.

Protocolo

Descripción formal de formatos de mensaje y las reglas de cómo dos computadoras deben intercambiar mensajes. Los protocolos describen detalles de interfaces de computadora a computadora de bajo nivel o intercambios de mensaje de alto nivel entre programas.

Puerto

Abstracción que los protocolos de transporte utilizan para distinguir entre muchos destinos en un host remoto. Los puertos son asociados a una aplicación específica y sea a través de este puerto que la aplicación recibirá peticiones de servicio, un claro servicio de esto es el servicio de TELNET al cual está asociado el puerto 21.

Query

1. Una petición para recuperar datos con el comando select.
2. Cualquier comando SQL que manipula datos.

Recuperación

Capacidad de restaurar la integridad y la consistencia de una B. D., después de una falla en el sistema.

Redundancia

Es la repetición de datos, datos derivados, por lo que genera inconsistencia.

Registro

Grupo de campos (columnas) cuya información se trata como una unidad. Equivalente lógico de un renglón.

Reglas

Las reglas pueden especificar una máscara para una columna o un tipo definido por un usuario. Una tabla puede ser una lista de valores, un rango o un patrón; una regla puede ser aplicada para varias columnas de una o distintas tablas.

Relación

Dada una serie de conjuntos D1, D2,..., Dn (no necesariamente distintos) se dice que R es una relación sobre estos n conjuntos si es un conjunto de n tuplas ordenadas <d1, d2,..., dn>, tales que d1 pertenece a D1, d2 pertenece a D2,..., dn pertenece a Dn. Los conjuntos D1, D2,, Dn son los dominios de R. El valor n es el grado de R.

Root

El login name convencional para el superusuario de Unix, el root es el único usuario de un sistema que posee ilimitado poder sobre el sistema.

Segunda forma normal

Una relación está en segunda forma normal si está en primera forma normal y cada atributo que no forma parte de la llave principal está en forma total y funcionalmente dependiente de ella.

Seguridad

Protección de los datos contra accesos, modificaciones o pérdidas.

Sesiones remotas

Conexión con sistemas remotos encargados del procesamiento desde cualquier computadora conectada a la red.

Software

Parte de un sistema computacional que se refiere a los programas que se utilicen para que funcione.

SQL

Structured Query Language (SQL), el lenguaje usado para comunicarse con una base de datos relacional.

Tabla

Una colección de renglones (registros) que tienen asociadas columnas (campos). El equivalente lógico de un archivo de base de datos.

TCP

Acrónimo de Transmission Control Protocol, uno de los protocolos primarios usados en Internet. TCP permite a un proceso en una máquina enviar un flujo de datos a otro proceso en otra máquina. Para realizar una conexión TCP los participantes deben establecer una conexión antes de enviar información.

Tercera forma normal

Una relación está en tercera forma normal si está en segunda forma normal y ningún atributo involucrado en la relación es funcionalmente dependiente de algún otro atributo que no es parte de la llave.

Tipo de dato

Especifica que tipo de información debe tener cada columna, y como el dato debe ser almacenado. Los tipos de datos incluyen, char, int, money, y todos los demás. Los usuarios pueden construir sus propios tipos de datos.

Trade_offs

A mayor integridad y consistencia mayor eficiencia.
A mayor seguridad menor eficiencia.
A mayor seguridad menor concurrencia.

Transact SQL

El lenguaje SQL usado por el servidor.

Trigger

Una forma especial de procedimiento almacenado la cual tiene efecto cuando un usuario utiliza comandos tales como: insert, delete o update para determinadas tablas o columnas. Los triggers son usados a menudo para lograr la integridad referencial.

Workstations

Estación de trabajo, equipos de cómputo de gran desempeño con capacidad de multiprocesamiento y acceso multiusuario, estos sistemas fueron inicialmente desarrollados para efectuar tareas propias de ingeniería, en la actualidad estos sistemas han venido a sustituir a equipos como minicomputadoras y mainframes.

Variable

Estructura de almacenamiento temporal que recibe un nombre único asignado por el programador.

Vista

Un comando llamado select que almacena la base de datos como un objeto. Permite a los usuarios ver un conjunto de columnas o renglones de una o más tablas.

BIBLIOGRAFÍA

Aprendiendo HTML para Web en una semana
Laura Lemay
Sams Publishing
México, 1995.
Pp. 1-181, 205-377.

CGI Programming
Dan Berlin
Sams net
E. U. A. 1996.
Indianapolis, Indiana
Capítulos 1, 2, 3, 4, 9 y 10.

JavaScript 1.1, Developer's Guide
Arman Danesh, Wes Tatters.
Editorial Sams net.
Indianapolis, Indiana.
Pp. 3- 47, 131-162

Manual de referencia deHTML.
Todos los temas.
México D. F., 1997.

PERL 5
Kamran Husain
Sams Publishing
E. U. A. 1996.
Indianapolis, Indiana 46290.
Pp. 4-46, 96-117, 136-160, 406-431.

PERL: Páginas Web interactivas
Juan Palacio
Editoral ra-ma
México, 1998.
Todos los temas.

Sybase Developer's Guide
Daniel J. Worden.
Editorial SAMS Publishing, 1 edición.
E.U.A. 1996.
Pp. 3-27, 49-123, 133-221, 255-285.

SYBASE SQLServer 11.
Ray Rankins, Jeffrey R. Garbus, David Solomon, Bennet Wm. McEwan.
Sams Publishing.
E. U.A., 1996, 1 edición.
Indianapolis, IN 46290.
Págs. 3-22.

Sybase web.sql Programer's Guide.
Document ID:35725-01-0101-02.
Sybase, Inc. All rigths reserved.
Todos los temas.

UNIX (Manual de referencia)
Stephen Coffin
Editorial Mc Graw Hill
5° edición
Pp. 5-18.

Web Site Administrator's (Survival guide)
Jerry Alban y Scott Yanoff
Editorial Sams Net
Indianapolis, Indiana
Pp. 3-25.

Worl Wide Web
John December y Nel Randall
Editorial Sams Publishing
Indianapolis, Indiana
Pp. 3-275, 283-439.

Worl Wide Web. Database Developer's Guide
Swank Mark and Drew Kittel
Editorial Sams_Net
E.U.A. 1996.
Pp. 3-27, 49-123, 133-121, 255-285.

Direcciones electrónicas:
http://www.apache.org/
http://language.perl.com/info/documentation.html
http://macaria.pucp.edu.pe/taller_html/URL-General.html
http://www.sybase.com/