UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE INFORMACION BAJO EL CONCEPTO WORLD WIDE WEB CON SYBASE UTILIZANDO NETSCAPE Y POWERBUILDER COMO FRONT-ENDS

T E S I S

PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION
P R E S E N T A N :

YOSELINDA AGUIRRE MIYASAKI
JUAN BAUTISTA MARTINEZ
ANA CLAUDIA DIAZ MANZANARES

DIRECTOR DE TESIS: ING. DOMINGO PALAO MUÑOZ

CIUDAD DE MEXICO, MAYO DE 1997

Mi eterno agradecimiento...

"La dedicación conquista todas las cosas, excepto un dolor de muelas"

A mis padres por su apoyo y consejos durante todos estos años, y como muestra de que todos los sacrificios hechos no fueron en vano.

A mis hermanos Oscar, Jorge, Lety, Viri, a mi cuñado Beto y a mi sobrino Miguelito por hacer mi existencia más agradable.

Por su comprensión un beso a Reina.

Un reconocimiento a la Máxima Casa de Estudios: la Universidad Nacional Autónoma de México, y en especial a la Facultad de Ingeniería que junto con sus profesores forman profesionistas con un espíritu universitario.

Mi agradecimiento a la Dirección General de Asuntos del Personal Académico por las facilidades otorgadas para la realización de este trabajo, en particular al Dr. José Luis Boldú Olaizola y a la Mtra. Estela Morales.

Por su confianza al M. en I. Octavio Estrada y a la Lic. Margarita Ramirez.

Por los conocimientos enseñados como profesor y por la vocación cultivada como profesionista al Ing. Domingo Palao, mi director de tesis.

Por ampliar mis dudas, no es cierto!, por su compañía y apoyo durante los últimos tres años a mis compañeros de trabajo del Departamento de Sistemas de la Dirección General de Asuntos del Personal Académico: Ale, Claudia, Arturo, Rosaura, Rosario, Ricardo, Héctor, Juan Francisco, Gaby, David, Lupita, Bety, Roman, Yose, Rafael y Judy.

Un abrazo muy grande para mis compañeras de tesis Yose y Claudia.

Para mis mejores amigos: Manuel, Marco, Gabriel, Jaime y Arturo.

"... Y el conocimiento hizo la diferencia"

Sinceramente

Juan Bautista Martínez

AGRADECIMIENTOS.

Doy gracias a Dios por permitirme concluír este trabajo de tesis, a mis padres por haberme brindado la oportunidad de prepararme y darme con ello las herramientas para enfrentar esta vida, agradezco también a todas aquellas personas que de manera incondicional de alguna forma contruyeron para su realización, y finalmente a mis compañeros de tesis, por su infita paciencia y tolerancia. A todos ellos mil gracias.

Ana Claudia Diaz Manzanares.

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México que me dió la oportunidad de tener una formación profesional.

A Raúl Aguirre y Ma. del Carmen Miyazaki

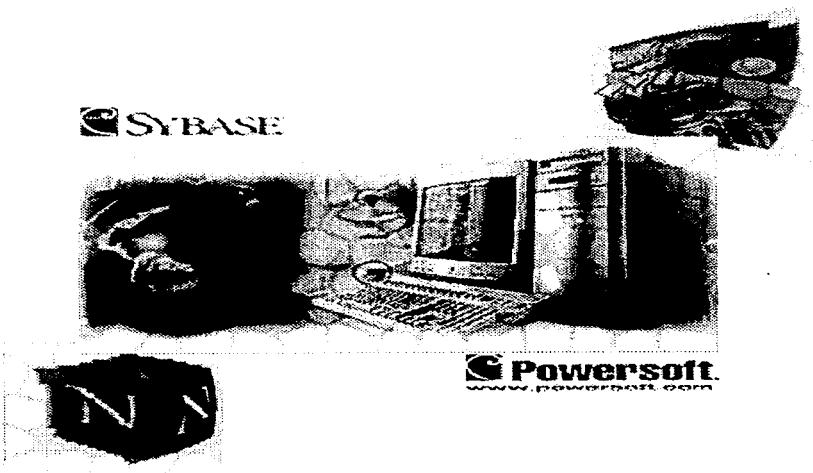
A ustedes les dedico este trabajo agradeciendoles todo el apoyo y la oportunidad que me dieron para subir a un peldaño más en mi vida. Gracias, por el ejemplo y todas las cosas buenas y malas que me dan para salir adelante y continuar desarrollandome.

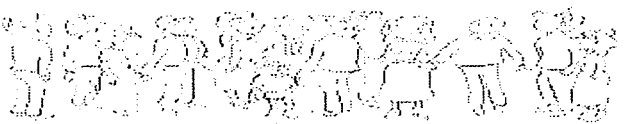
Agredezco a toda la gente que contribuyo con la realización de esta tesis pero con especial énfasis a Juan Bautista de quien aprendí muchas cosas y a quien aprecio mucho.

Yoselinda

Indice

1. Introducción





La Dirección General de Asuntos del Personal Académico (DGAPA) es una institución administrativa universitaria que depende de la Secretaría General de la UNAM y tiene la tarea de administrar los programas de fortalecimiento, apoyo y superación del personal académico. En los años 80's pensar en equipo XT y terminales A12 que resolvieran su proceso de sistematización de datos era concebible.

A partir de los años 90's el crecimiento de la DGAPA fue sensible año tras año, debido al aumento de los programas académicos creados por el Rector y la Secretaría General. Uno de estos programas fue el PAPIID (Programa de Apoyo a Proyectos de Investigación e Innovación Docente) creado en 1989 y posteriormente denominado PAPIIT (Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica) cuyo objetivo es apoyar a profesores e investigadores de carrera de la UNAM en sus proyectos de investigación. El PAPIIT se compone de tres etapas: solicitud de ingreso, solicitud de renovación e informe final.

❖ Requerimientos del Sistema

El proyecto que se presenta como tema de tesis surge como una necesidad en la DGAPA de ofrecer una mejor alternativa a los usuarios de dicho programa en el proceso de inscripción. Uno de los objetivos es sustituir paulatinamente la forma tradicional del manejo de la información en discos flexibles que consiste de dos fases: en la primera, la DGAPA entrega a cada investigador un disco flexible con un sistema de captura (desarrollado en Clipper) cuya finalidad es ayudar al investigador a requisitar su solicitud a través de archivos en donde se almacena la información referente al proyecto de investigación con el que participará; en la segunda fase el investigador devuelve el disco a la DGAPA, una vez terminada la captura de los datos. Esta forma de manejar la información presenta algunas desventajas como son:

pérdida de información a causa de ***virus informáticos***, pérdida de tiempo en recoger el sistema de captura, regresar el disco con la información capturada y en caso de que el sistema de captura tuviera alguna falla se tendría que recoger nuevamente un disco con el sistema de captura ya corregido.

Virus informático: es un conjunto de programas que dañan el funcionamiento de un sistema o equipo de cómputo.

Una desventaja más que se presenta es la incongruencia y redundancia en los datos entre las tres etapas que se manejan en PAPIIT, lo cual origina una pérdida parcial del control de la información.

Otro objetivo del proyecto es permitir la captura en línea, mediante un sistema en red, de esta manera que el servicio sea flexible y más rápido. Flexible, porque la captura de datos podría hacerse en distintos lugares, sin la necesidad de instalar el sistema localmente y con la posibilidad de ingresar los datos paralelamente (desde dos o más máquinas simultáneamente).

Más rápido, por que el investigador ya no tendría que invertir tiempo en recoger y enviar su disco de captura, además, de que si existiera una falla en el sistema, los investigadores no tendrían que detener su trabajo para esperar una solución al problema, sino que ellos continuarían trabajando en otros apartados del programa en lo que se realiza la corrección de la falla por parte del Departamento a cargo de la programación del sistema.

Además, se tendrían beneficios tales como: reducción de gastos por parte de la dependencia, en discos, hojas, tinta, disminución de los trámites administrativos y, así mismo un control de los datos concentrados en una misma base de datos.

Internet
Es conjunto de computadoras conectadas mundialmente.

Operativamente, el personal de la UNAM dedicado a la investigación podrá acceder el sistema desde cualquier parte del mundo con solo tener una computadora que posea una conexión a la ***red Internet***, un software

que visualice ***hipertexto*** y solicitar su clave de acceso al sistema, al ingresar por primera vez a éste.

Uno de los puntos importantes es brindar transparencia al usuario, de tal manera que no se preocupe por el procedimiento que tiene que realizarse para el almacenamiento y presentación de la información capturada.

Para el usuario, la facilidad en la captura de información es un punto muy importante, nuestro proyecto pretende usar una interface gráfica que ofrezca sencillez, agilidad y comodidad en la captura de la información evitando así que el usuario no se encuentre en un mundo de teclas y comandos difíciles.

La información capturada por los usuarios será almacenada en una base de datos que se encontrará, al igual que el sistema, en el servidor localizado en la DGAPA, por lo que los investigadores no tendrán que preocuparse por espacio de disco necesario para el resguardo de su información.

Finalmente, como parte fundamental de las características de nuestro proyecto esta la cobertura de distintas plataformas de equipos de cómputo como son: Windows, Macintosh y X-Windows; de está manera, el investigador tendrá diferentes opciones para usar una máquina en la captura de sus datos.


Hipertexto: Es un texto en el cual determinadas palabras o frases derivan a un nuevo documento relacionado con el contenido de dicha palabra o frase.


Visualizador o Browser: Programa interactivo usado para acceder información en Internet.


HTML: Es lenguaje usado para escribir documentos en hipertexto.

➢ Definición del proyecto

Los grandes avances tecnológicos en redes y hardware han sido base del surgimiento de nuevos conceptos en el área de software; como ejemplo, tomemos a los ***vizualizadores o browsers.***

Los visualizadores o browsers utilizan un lenguaje de hipertexto conocido como HyperText Markup Language (***HTML***), el cual forma parte

de la tecnología ***World Wide Web, WWW, W3 o Web*** que hace posible el acceso a millones de páginas(archivos) de información; éste consiste en una serie de definiciones estándar que le indican a la página si el contenido es texto, gráfica, sonido o vídeo. El lenguaje de hipertexto e hipermedia es el motor central de lo que hoy se conoce como World Wide Web.

Un vizualizador permite al usuario ***"navegar"*** a través de las páginas de hipertexto disponibles en el Web.

Las computadoras que proporcionan información son llamadas servidores Web y utilizan un protocolo de transferencia de hipertexto conocido como HyperText Transfer Protocol (***HTTP***). HTTP le permite a los servidores Web aprovechar la tecnología del hipertexto para enlazar de manera conjunta documentos y buscar información, implementando así un sistema de "navegación" en hipertexto, permitiéndole a los usuarios moverse libremente de un documento a otro sobre la red, independientemente del lugar donde estos se encuentren localizados o del equipo de cómputo que estén utilizando para acceder al servidor, como puede ser una computadora personal (PC) o una estación de trabajo (WorkStation).

Históricamente Mosaic fue el programa que aportó más conceptos e ideas a los actuales visualizadores o browsers. Mosaic se liberó al público, sin costo alguno, en el año de 1993. La facilidad de uso de Mosaic hacía mucho más rica y gratificante la "navegación" en el Web para cualquier usuario. Se trataba pues, de estar prácticamente hojeando una nueva forma de publicación en tiempo real con textos de diferentes tamaños y colores, elementos gráficos, fotografías a todo color y hasta pequeñas secuencias de vídeo por computadora.

El código fuente de Mosaic fue reprogramado y mejorado, y se le llamó Netscape. Netscape surgió como un software de dominio público, hoy en día es comercial, sin embargo existe una versión de dominio público en la red Internet. Netscape es fácil de usar y de instalar, permite comunicar


WWW: Servidor de información, desarrollado en el CERN (Laboratorio Europeo de Física de Partículas), con el fin de construir un sistema distribuido de hipermedia e hipertexto.


Navegar
Se refiere al hecho de pasar de un documento a otro dentro del Web.


HTTP:
es un protocolo con la ligereza y velocidad necesaria para distribuir y manejar sistemas de información hipermedia.

distintas computadoras conectadas en red, captura de datos y tiene disponibles versiones para Macintosh, Windows y Unix. Estas características lo hacen el visualizador más popular del momento y por ello nos inclinamos a usar a Netscape como front-end.

Netscape sería el medio de comunicación disponible para los usuarios del sistema, y el medio de almacenamiento y manipulación de datos se realizaría a través del Sistema Manejador de Bases de Datos (DBMS) Sybase.

Por qué usar Sybase y no otro de los tantos DBMS's que existen actualmente como lo son: Oracle, Informix, Ingres o Proggress ?. La respuesta es sencilla, primero tomemos en cuenta que Sybase es el estándar que se adoptó en la Universidad Nacional Autónoma de México (UNAM), por tanto es el software que se adquirió en la DGAPA; segundo, históricamente Sybase abrió caminos en la evolución de mainframes a Cliente/Servidor con la introducción de SYBASE®SQL Server en 1987, siendo este el primer sistema manejador de bases de datos relaciónales (RDBMS) diseñado específicamente para sistemas abiertos de multiprocesos simétricos esto gracias a su arquitectura VSA(Virtual Server Architecture) lo que le permite obtener un incremento en el desempeño del 30% al 50% sobre versiones normales de UNIX, soporta más de 30 versiones de UNIX, soporta hasta 1,000 usuarios simultáneos de la base de datos, el lenguaje nativo de SQL Server es Transac SQL; además, Sybase ofrece otras herramientas, corriendo bajo UNIX, que utilizan el lenguaje APT-SQL, un SQL adicionado con elementos 4GL, para acelerar el desarrollo de aplicaciones, ofrece soporte a demandas complejas de aplicaciones de cómputo que requieran un procesamiento de transacción en línea intensivo, velocidad y confiabilidad. Estos puntos hacen que elijamos a Sybase como otra de las herramientas a usar en nuestro proyecto.

Con Netscape y Sybase, como herramientas, podríamos presentar un

sistema de captura en línea que permita a los investigadores de la UNAM, capturar su información de las diferentes etapas del PAPIIT.

Ahora, para cubrir las necesidades internas de la dependencia se necesita de un programa que permita consultar la información, generar reportes e inclusive generar gráficas a partir de la información capturada por los investigadores.

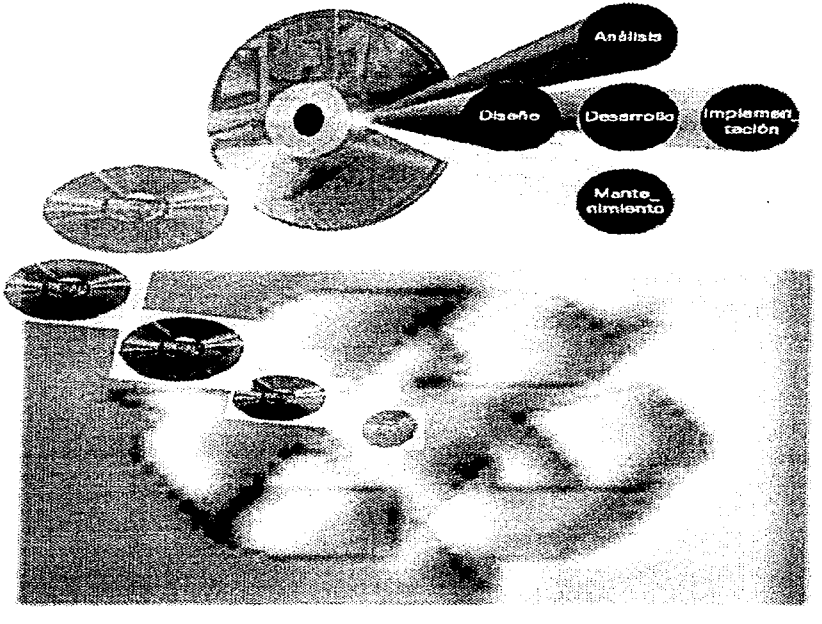
Actualmente se ofrecen múltiples alternativas en el desarrollo de sistemas, como vimos Netscape es una muy buena opción sin embargo no podría satisfacer las necesidades internas de la dependencia, por lo que decidimos utilizar PowerBuilder como otro front-end que cubriera en su totalidad estas demandas.

PowerBuilder es una poderosa herramienta de desarrollo de sistemas basada en GUI's.

Históricamente las primeras GUI's tienen origen en los años 70's y fueron desarrolladas por SRI y Xerox, posteriormente Apple fue quien dio un gran auge a las GUI's con la introducción de la Macintosh en 1980, la cual estaba provista de un sistema operativo muy sencillo y fácilmente entendible por cualquier persona. En este sistema operativo los comandos que puede ejecutar el usuario no tienen que ser tecleados, sino que son realizados con la ayuda de una interface basada en gráficos y un dispositivo de señalamiento (puntero o mouse). Así, por ejemplo, para copiar un archivo el usuario sólo tiene que "arrastrar" al archivo, representado por un imagen, del origen al destino.

Nuestro proyecto tiene el objetivo de dar un mejor servicio a la planta académica de la UNAM y para ello utilizaremos las siguientes herramientas: Netscape y PowerBuilder como front-end y Sybase para la manipulación y el almacenamiento de datos.

2 Ciclo de Vida



Para tratar este tema es necesario iniciar con la pregunta ¿qué es el software?. Existe una variedad de respuestas, por ejemplo la siguiente: "instrucciones en programas de computadora que cuando se ejecutan proporcionan la función y el comportamiento deseado". Este concepto será más claro si ampliamos nuestro conocimiento y entendemos como ha evolucionado el software, y el porqué de su importancia.

El software

El software ha tratado de evolucionar a la par que el ***hardware***, sin embargo, en un inicio existían métodos poco sistemáticos para programar; siendo los sistemas de ***propósito general*** y de ***uso dedicado***, es decir, generalmente el lenguaje utilizado era aplicado para la realización de programas monitores y una sola persona escribía, ejecutaba y depuraba los programas. Por lo anterior, no era posible comercializar el software como en estos tiempos.

Cuando se introducen los ***sistemas multiusuarios*** con ***técnicas interactivas*** y los dispositivos de almacenamiento, se generan los primeros sistemas de gestión de bases de datos. Es en este momento donde el software se comercializa, en el amplio sentido de la palabra, y aparece el "mantenimiento" del software para la corrección de fallas futuras, en un sentido metódico y más estructurado. En ese período el software se personaliza tanto, que con las nuevas exigencias que traía consigo la aparición de las redes de computadora y el avance siempre más acelerado del hardware en relación al software, se provoca una "crisis" del software que se desata a mediados de los años setenta. Posteriormente, con el uso tan amplio de los microprocesadores en aparatos electrónicos, se da pie a que el software sea requerido no sólo en sistemas de cómputo; además, la venta de este para sistemas personales se diversifica a tal grado que su costo se incrementa considerablemente.


Hardware: Se refiere al equipo y maquinaria de un equipo de cómputo.


Programas de propósito general: Son aquellos que resuelven una amplia variedad de problemas.


Sistemas multiusuarios: Son sistemas en los que un computadora atiende a varios usuarios a la vez.


Técnicas interactivas: permiten un diálogo bilateral entre el usuario y un equipo de cómputo.

Actualmente, hacemos uso de una amplia gama de conocimiento y de herramientas, lo que ha abierto grandes posibilidades para el reconocimiento de formas y procesamiento de información. A pesar de todos estos avances, la crisis de que antes se habló todavía no se encuentra totalmente sufragada.

Características del software

Ahora bien, ya que conocemos a grandes rasgos el proceso evolutivo de software, podemos hablar un poco de sus características y sus componentes.

Debido a que el software es un elemento lógico del sistema posee características muy particulares: a) el software se desarrolla, no es algo que se pueda fabricar; b) el software no se descompone, pues no es susceptible a cambios en el entorno físico; sin embargo sí se deteriora, debido a los defectos no detectados que harán que falle el programa, a un error en el diseño o en la codificación, y para lo cual no existen partes de repuesto; c) el software se maneja como una unidad completa, y aunque esta es la manera en que se ha venido trabajando, se está incursionando en algo que podríamos llamar *"ensamble de software"*, es decir, actualmente pequeños módulos de un programa son reutilizados en otros sin necesidad de ser recodificados (a esto se le conoce como "reusabilidad" del software).

Ahondando un poco en la última característica mencionada, la "reusabilidad" se ha tratado de incorporar al mundo del cómputo desde hace algunos años. En un inicio se crearon ***bibliotecas*** que contenían únicamente algoritmos bien definidos, pero que no dejaban de ser generales. Actualmente se encapsulan datos y detalles de procesamiento en un único paquete, llamado objeto, permitiendo incorporarlo como elemento a nuevas aplicaciones, pues están contenidos en bibliotecas orientadas a la construcción de interfaces.

Biblioteca: es una colección de funciones o subrutinas prescri-tas que se unen a un programa principal cuando este se compila

Componentes del software y lenguajes de programación

Los componentes del software se refieren a cada una de las partes del mismo que hacen corresponder los requisitos del cliente con una aplicación específica. Dichos componentes se construyen mediante un ***lenguaje de programación*** que tiene un vocabulario limitado, una gramática y reglas bien definidas de sintaxis y semántica. Como vemos, el lenguaje de programación es una herramienta esencial para el desarrollo de cualquier software, sin embargo no existe un lenguaje único para programar, hay lenguajes enfocados a áreas específicas del conocimiento, e incluso algunos de ellos requieren para su uso de una ***arquitectura*** o un ***sistema operativo*** específicos. Entre las clases de lenguajes utilizados se encuentran:

Lenguajes máquina. Consiste en una representación simbólica del conjunto de instrucciones de la ***CPU***, por ello para programar en él se requiere de un excelente conocimiento de la arquitectura de la CPU.

Lenguajes de alto nivel. Permiten al programador independizarse del pleno conocimiento de la arquitectura de la máquina y sólo usan instrucciones entendibles al programador y después traducidos a lenguaje máquina.

Los dos tipos de lenguajes anteriores se consideran "lenguajes procedimentales", pues el programador tiene que enfocarse tanto a la especificación de la estructura de la información como al control del propio programa, es decir, es necesario, especificar la acción para conseguir un resultado deseado. La contraparte la encontramos con los "lenguajes no procedimentales" o de "cuarta generación", pues en ellos se especifica únicamente el resultado deseado y no los detalles procedimentales (entre ellos se encuentran las aplicaciones para bases de datos).

Lenguaje de Programación: Conjunto de instrucciones que nos permiten desarrollar aplicaciones.

Arquitectura: Determina el estándar para los dispositivos que se conectan y para el software que se ejecutar en un equipo.

CPU. Es la unidad Central de procesamiento de una computadora.

Ingeniería del software

Ahora bien, no pensemos que el desarrollo del software se limita a características y componentes, sino que comprende toda una disciplina, la

cual es llamada ***Ingeniería del software***, que se vale de métodos para su desarrollo, de herramientas para automatizar estos métodos mencionados, y garantizar así una forma de implementación, que permita durante el desarrollo, la coordinación, control y gestión. Existen tres elementos clave que facilitan el control del proceso de desarrollo del software y son *métodos*, *herramientas* y *procedimientos*. Dichos elementos constituyen algo que se denomina ***paradigmas del software*** y que explicaremos cada uno a continuación:


Ingeniería del software: Diseño, documentación y desarrollo de software.


Paradigmas del Software: Metodologías y herramientas existentes para el desarrollo de software.


Arquitectura de programas: Comprende la estructura de programación que poseerá uno o varios programas.

Método. Indica cómo construir técnicamente un software e incluye planificación y estimación de proyectos, análisis de los requisitos del sistema, diseño de estructuras de datos, ***arquitectura de programas*** y procedimientos algorítmicos, codificación, prueba y mantenimiento.

Herramientas. Suministran un soporte automático o semiautomático para aplicar los métodos. En la actualidad existen herramientas capaces de generar información de utilidad para otras herramientas, combinando así software, hardware y bases de datos sobre ingeniería de software.

Procedimientos. Fusiona los métodos con las herramientas, pues definen la secuencia en la que se aplican los métodos, la realización de informes, formas, etc., los controles de desarrollo y la coordinación de los cambios y directrices que ayuden a los realizadores de software a evaluar el progreso en el desarrollo.

La elección de los pasos que abarcan los métodos, las herramientas y los procedimientos antes mencionados, se llevan a cabo de acuerdo con la naturaleza de un proyecto y de la aplicación misma. De aquí, que sea necesario, en ocasiones, la combinación de ellos.

➢ Ciclo de vida de un sistema de cómputo

Un software se desarrolla con base a paradigmas de programación. Los paradigmas del software más discutidos y conocidos son el *ciclo de vida clásico*, *construcción de prototipos* y el *modelo en espiral*. A continuación

daremos una breve explicación de cada uno de ellos.

Ciclo de vida clásico

El ciclo de vida clásico exige un enfoque sistemático y secuencial del desarrollo del software que progresa a través del análisis, diseño, codificación, prueba y mantenimiento. Las etapas que abarca son:

Ingeniería y análisis del sistema: Aquí se establecen los requisitos generales de todos los elementos de que consistirá el sistema y luego se asigna algún subconjunto de estos requisitos al software. Este planteamiento del sistema es esencial cuando el software debe interrelacionarse con otros elementos, tales como hardware, personas y bases de datos.

Análisis de los requisitos del software: Se realiza una recopilación intensa sobre los requisitos, centrándose especialmente en los más necesarios para el software.

Interfaz: hace referencia a una conexión o interacción entre hardware, software y/o usuario.

Diseño: Se enfoca sobre cuatro atributos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la ***interfaz***. El proceso de diseño traduce los requisitos en una representación del software que puede ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación.

Codificación: El diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

Prueba: La prueba se centra en la lógica interna del software, asegurando que todas las sentencias sean probadas; en las funciones externas, se realizan pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

Mantenimiento: El software, indudablemente, sufrirá cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo, debido a que el cliente requiera ampliaciones funcionales o de rendimiento. El mantenimiento del software aplica cada uno de los pasos

precedentes del ciclo de vida de un programa existente en vez de a uno nuevo.

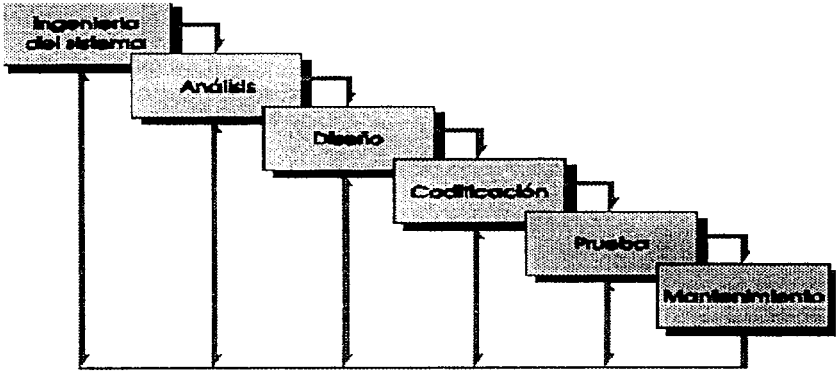


Fig. Ciclo de vida clásico

Construcción de prototipos

La construcción de prototipos es un proceso que facilita al programador la creación de un modelo de software a construir. El modelo tomará una de las tres formas siguientes:

1. Un prototipo en papel que describa la interacción hombre-máquina, de forma que facilite al usuario la comprensión de cómo se producirá tal interacción.
2. Un prototipo que implemente algunas funciones requeridas del programa deseado.
3. Un programa existente que ejecute parte o toda la función deseada.

Como en todos los métodos de desarrollo del software, la construcción de prototipos comienza con la recolección de los requisitos, seguido de un diseño rápido que se enfoque sobre la presentación de los aspectos del software visibles al usuario, posteriormente se construye el

prototipo, el cual es evaluado por el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar.

Se produce un proceso interactivo en el que el prototipo es afinado para que satisfaga las necesidades del cliente, al mismo tiempo que facilita a quien lo desarrolla una mejor comprensión de lo que hay que hacer. Por lo tanto, el prototipo es un mecanismo cuya función principal es identificar los requisitos del software.

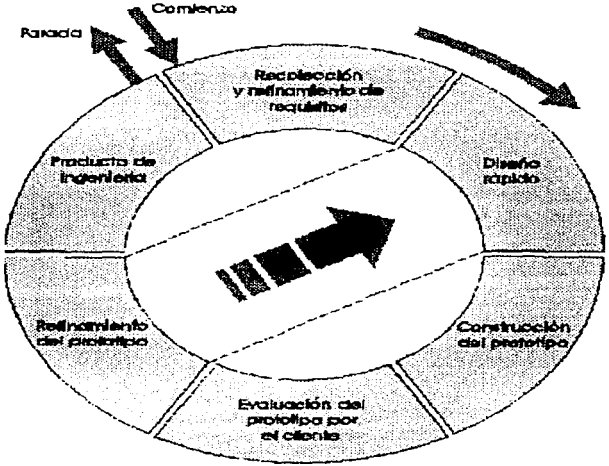


Fig. Creación de Prototipos

Modelo en espiral

El modelo en espiral ha sido desarrollado para cubrir la mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo un nuevo elemento: el análisis de riesgo que no incluye ninguno de los modelos antes mencionados.

Este modelo define cuatro actividades principales:

1. Planificación: determinación de objetivos, alternativas y restricciones.
2. Análisis de riesgo: análisis de alternativas e identificación/resolución de riesgos.
3. Ingeniería: desarrollo del producto.
4. Evaluación del cliente: valoración de los resultados de la ingeniería.

Un aspecto importante del modelo en espiral se hace evidente cuando consideramos la dimensión radial representada en la figura abajo mostrada.

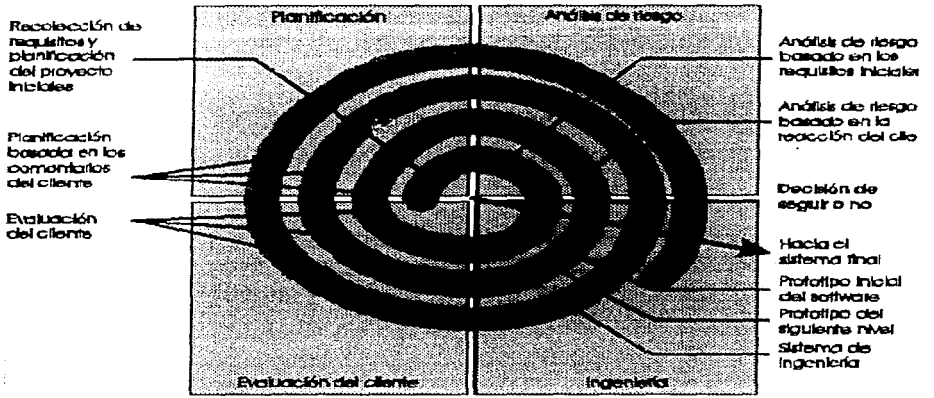


Fig. Modelo en Espiral

Con cada iteración alrededor de la espiral se construyen versiones sucesivas del software, cada vez más completas. Durante la primera vuelta alrededor del espiral se definen los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. - Si indica la existencia de un riesgo se hace uso de un prototipo. El cliente evaluá el trabajo y

sugiere modificaciones.

Con base en los comentarios del cliente se produce la siguiente fase de planificación y de análisis de riesgo. Si los riesgos son demasiado grandes, se puede dar por terminado el proyecto. Sin embargo en la mayoría de las veces se sigue avanzando alrededor del espiral, y ese camino lleva a los desarrolladores hacia fuera, hacia un modelo más completo del sistema y, al final, al propio sistema operacional.

Técnicas de Cuarta Generación

El paradigma de cuarta generación no se mencionó como uno de los más conocidos, sin embargo hemos abierto un espacio para hablar de él, debido a la importancia que esta cobrando en nuestros días.

El paradigma *Técnicas de Cuarta Generación* (T4G) para la ingeniería de software se orienta hacia la posibilidad de especificar el software a un nivel más próximo al lenguaje natural o una notación que proporcione funciones significativas. Algunas de las herramientas que puede incluir son: lenguajes no procedurales para consulta a bases de datos, generación de informes, manipulación de datos, interacción y definición de pantallas, generación de código, facilidades gráficas de alto nivel y facilidades de hoja de cálculo

. Al igual que en otros modelos este modelo comienza con la recolección de requisitos. Idealmente, el cliente describe los requisitos, que son a continuación, traducidos directamente a un prototipo operativo. Para aplicaciones pequeñas, se puede ir directamente desde el paso de recolección de requisitos al paso de implementación, usando un lenguaje de cuarta generación no procedimental. Sin embargo, es necesario un mayor esfuerzo para desarrollar una estrategia de diseño para el sistema. Una característica importante es que la implementación se realiza con un ***Lenguaje de cuarta generación*** (L4G), lo que permite centrarse en la representación de los resultados deseados. Finalmente, el desarrollo se

Dentro del proceso evolutivo de los lenguajes de programación, el lenguaje de cuarta generación abarca esta caracterización, porque no requieren de la lógica tradicional entrada-proceso-salida..

realiza pensando en que debe ser construido de tal forma que facilite el mantenimiento.

Paradigma del Proyecto

Nuestro proyecto utiliza como paradigma del ciclo de vida una combinación de los modelos "ciclo de vida clásico" y "construcción de prototipos". La razón para elegir dichos modelos es consecuencia de la forma de trabajo de la dependencia, en donde se desarrolla el proyecto.

El modelo resultante sugerido para el desarrollo del sistema es el siguiente:

1.- Análisis del problema
2.- Recolección y refinamiento de requisitos
3.- Elaboración de prototipo inicial del sistema
4.- Prototipo final
5.- Planificación de actividades
6.- Diseño del sistema
7.- Desarrollo del sistema
8.- Pruebas al sistema
9.- Implementación
10.- Evaluación

El método para el desarrollo del sistema lo podemos observar en la siguiente figura.

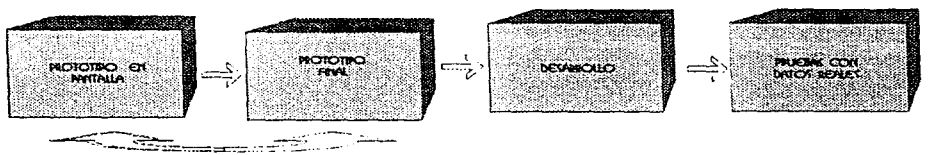


Fig. Método planteado para el desarrollo del sistema

El modelo propuesto pretende alcanzar una comunicación estrecha entre nosotros como desarrolladores y las personas encargadas de administrar el Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT). Ahora bien, explicaremos en que consiste cada uno de los puntos que, para el desarrollo, planteamos en el modelo:

1. Análisis del problema

   En el se llevarán a cabo entrevistas con el departamento a cargo de la administración del programa, para ver las posibilidades de desarrollo de un sistema en un ambiente de red que cubra las exigencias y necesidades de los usuarios hacia los cuales se dirigirá éste. Se planteará la posibilidad de desarrollar un sistema que explote el equipo y el software adquirido con anterioridad por la dependencia, que proporcione un servicio en lo posible eficiente, flexible y rápido.

2. Recolección y refinamiento de requisitos

   Para este punto llevaremos acabo el análisis para la creación de manuales de procedimiento, y documentos con el fin de comprender el manejo de la información. Así mismo, se pretende realizar entrevistas con el personal involucrado directamente en el manejo de la información del sistema, para determinar con ello, cuales serían los datos de entrada y salida, los que debe ser calculados y los que deben ser almacenados y/o respaldados. Estos ***respaldos*** se han pensado realizar con cierta periodicidad en cintas magnéticas debido a la importancia que ello demanda.

Respaldos: son copias de la información que se realizan por seguridad en un medio de almacenamiento diferente al lugar donde reside ésta.

3. Elaboración de prototipo inicial en pantalla del sistema

   Con base en la información obtenida en las dos etapas anteriores, se piensa llevar acabo un prototipo inicial, el cual representará toda la información señalada por las personas involucradas en el manejo administrativo del programa. En este prototipo se pretende realizar las primeras observaciones de la presentación de la información en

el sistema.

4. Prototipo final

   Será el resultado de la evaluación del prototipo inicial se creará un segundo prototipo que cubra las observaciones realizadas. Si es necesario se realizarán prototipos posteriores hasta que se considere se tienen claramente definidos los requerimientos.

5. Planificación de actividades

6. Tomando en cuenta las fechas especificadas por el departamento a cargo de la administración del programa, se estimarán los tiempos de: diseño, desarrollo, presentación, conformidad, pruebas y modificaciones posteriores. Además, se asignará un tiempo de holgura a cada actividad. Esta planeación se sustentará en ***diagramas de GANT***, que esquematicen el tiempo (semanas) en el que se deberán realizar todas y cada una de las actividades.

Diagramas de Gant: Gráfica de barras,, donde en el eje horizontal se maneja el tiempo mientras que en el eje vertical, las actividades involucradas en un proceso.

7. Diseño del sistema

   Una vez aprobado el prototipo final así como el diagrama de tiempos, se procederá al diseño de la base de datos (homologación de los campos en cuanto a nombres y tamaño), diseño de programas (arquitectura del programa), programas a usar, ubicación física de los datos (bases de datos, tablas, vistas, etc.), residencia de los programas (fuentes, ejecutables y librerías) e imágenes en los directorios correspondientes.

8. Desarrollo del sistema

   Se llevará acabo la creación de las bases de datos, tablas, vistas, dispositivos lógicos de respaldo, además del desarrollo de la programación. Es importante hacer notar que la información existente, por cuestiones de conveniencia y debido a que este sistema será una alternativa adicional proporcionada por el PAPIIT, tendrá que interactuar en un momento dado con y en las anteriores herramientas de trabajo, lo que implica un sistemas de conversión

de información de una plataforma a otra.

9.-Pruebas al sistema

Las pruebas del sistema se piensa puedan realizarse en dos fases principales: una primera etapa, que considera una revisión operativa y que sería llevada a cabo por nosotros como desarrolladores; y una segunda fase, que contempla una revisión de los requerimientos reales y que estará a cargo del departamento encargado de la administración del programa.

10.-Implementación

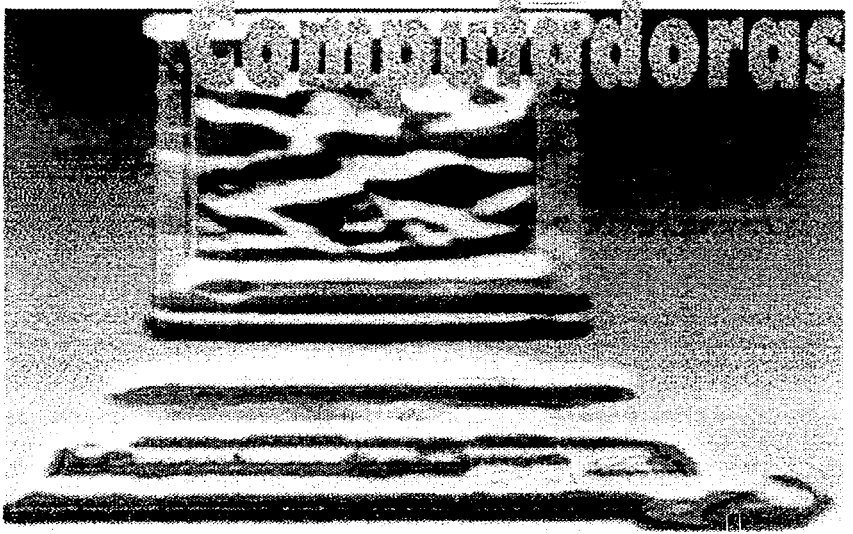
El sistema se liberará para su uso en la fecha indicada. Definiendo un horario de acceso, con la finalidad de realizar respaldos de la información capturada, y pensando también en que la comunicación de la red y el suministro eléctrico no esta a cargo del departamento donde se realiza este sistema.

11.-Evaluación

Esta se realizará en dos fases; en la primera cuando el sistema este en servicio, los usuarios nos darán a conocer sus puntos de vista acerca del funcionamiento del sistema por medio de correo electrónico y vía telefónica. En la segunda fase al cierre del sistema el personal usuario, aportará sus opiniones y sugerencias por escrito o a través del correo electrónico. Y finalmente, el otro indicador lo tomaremos del mismo desempeño que nosotros observemos en el sistema.

Es así como intentamos desarrollar el proyecto propuesto, acercando nuestra experiencia y nuestro conocimiento y dando flexibilidad para enfrentar cambios o requerimientos no contemplados.

3 Reflections



La necesidad de compartir recursos de cómputo, llevó a los diversos fabricantes y desarrolladores a idear redes. En un principio, las redes de computadoras se formaban por simples conexiones que permitían a un usuario acceder a recursos que se encontraban en otra computadora. En estas redes existían problemas de seguridad de los datos debido a que no se tenía control de quienes accedían la información.


File Server: Es un conjunto de programas que controlan la manipulación de archivos.

Hacia 1983, la compañía Novell introduce el concepto de ***file server*** (servidor de archivos) en el que todos los usuarios pueden tener acceso a la misma información, compartir archivos y contar con ciertos niveles de seguridad.

Los archivos y programas pueden accederse en modo multiusuario guardando el orden de actualización por el procedimiento de bloqueo de registros. Es decir, cuando algún usuario se encuentra actualizando un registro, se bloqueará éste para evitar que algún otro usuario lo extraiga o intente actualizar.

Las tendencias actuales indican una definitiva orientación hacia la conectividad de datos. No sólo el envío de información de una computadora a otra sino, en la distribución del procesamiento a lo largo de grandes redes en toda una empresa.

Fundamentalmente las nuevas clases de aplicaciones, los diferentes mecanismos de entrada/salida y la transferencia de información son parte de un sistema que combina la función de comunicación de datos, de transmisión de video y la función de comunicaciones por la voz en un sistema común llamado RED.

Tipos de redes

Redes de Área Local (LAN: Local Area Network)

Una LAN es una red de comunicaciones que permite la interconexión de una variedad de dispositivos de comunicaciones de datos en una área limitada.

Las LAN's son redes de alta velocidad usadas para interconectar computadoras en una pequeña área geográfica. El control de los dispositivos puede estar centralizado, distribuido o ser una combinación de ambos; trabajan a velocidades que pueden ir de 1 a 100 megabits por segundo. Transmite datos entre estaciones de usuarios, aunque algunas redes pueden transportar también imágenes y sonido.

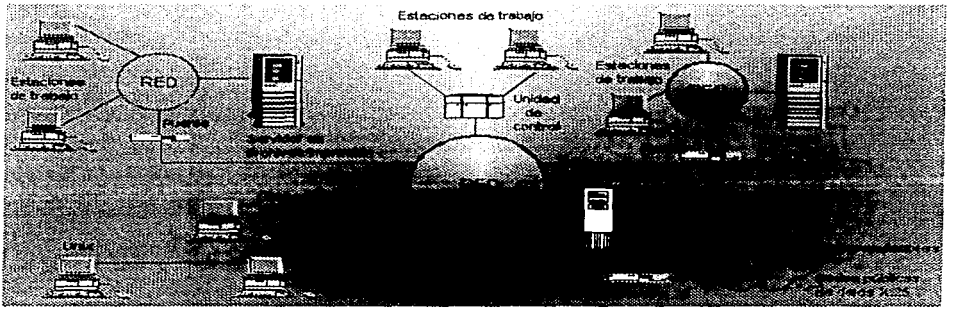


Fig. Red LAN

Una topología es la forma en que se interconectan las terminales o computadoras.

Las redes locales se apoyan en tres ***topologías*** principales para su configuración:

- Estrella
- Anillo y
- Bus

Los enlaces bidireccionales permiten la comunicación en dos sentidos.

- **Topología Estrella**

La conexión punto a punto comunica dos equipos de cómputo.

Es una de las estructuras más usadas en los sistemas de comunicación. Consta de una Unidad Central de Procesamiento (UCP) que controla el flujo de información a través de la red hasta todos los nodos, por medio de ***enlaces bidireccionales*** y son conectadas ***punto a punto***. Para transmitir un paquete, un ***nodo*** de la red lo manda al UCP, donde es posible tener varios esquemas de envío. El tamaño de la red está sujeto al poder

Un nodo es una terminal o computadora.

de la UCP, si el UCP se detiene, la red deja de funcionar. Pero el que un nodo de la red falle no repercute en el comportamiento global de la red y sólo se afectará el tráfico relacionado con ese nodo.

El hardware se refiere al equipo y maquinaria de un equipo de cómputo.

Ventajas:

1. La conexión del ***hardware*** es simple.
2. Presenta una gran flexibilidad para aumentar o disminuir el número de nodos debido a que estas modificaciones no representan ninguna alteración de su estructura y están localizadas en el nodo central.
3. Algunas líneas telefónicas existentes podrían servir como medio de transmisión.
4. Detección de errores en forma más aislada y fácil de corregirse.
5. Control descentralizado de todas las estaciones de trabajo.
6. Si un nodo deja de funcionar no altera el funcionamiento de la red.

Desventajas:

1. Usa más cableado, debido a que cada nodo debe ser conectada a su propio cable dedicado hacia el procesador central.
2. La instalación de un nodo adicional ocasiona dificultades, debido a que el cable debe instalarse desde el UCP.
3. Si el procesador central falla, la red deja de funcionar.
4. La comunicación entre nodo y nodo debe realizarse a través del servidor.
5. Debido a que la conexión es punto a punto es necesario utilizar más cable para dichas conexiones.

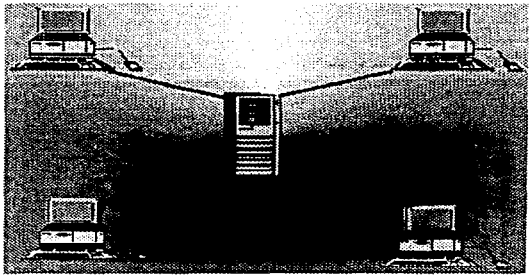


Fig. Topología Estrella

- **Topología Anillo**

En este tipo de redes, los nodos se encuentran unidos de manera que forman una configuración circular sin ninguna interrupción.

La red funciona a través de un enlace común en el que todos los nodos están conectados de manera lógica, y cada uno de ellos puede conectar o establecer comunicación con los demás por iniciativa propia, sin que ningún nodo ejerza algún tipo de control sobre él. Utiliza una técnica de acceso basada en la circulación de un patrón de bits único conocido como ***token***, el cual otorga el permiso de transmisión.

Un token es un paquete de información que viaja a través del anillo.

Normalmente en una red de anillo cada estación está conectada a la red a través de una interface especial, que es responsable de recibir, revisar y regenerar las señales que llegan a un nodo.

Las comunicaciones pueden circular en el anillo en ambas direcciones si las conexiones son ***Full-Duplex***. Sin embargo, en la mayoría de los casos, las redes de anillo son unidireccionales, ya que esto simplifica la interface. Esta tecnología fue adoptada en 1989 por el Institute of Electrical and Electronic Engeneers (IEEE) como IEEE 802.5 la cual describe el método de acceso y las especificaciones de la capa física para una red tipo Token Ring.

La conexión Full-Duplex permite la transmisión y recepción simultánea de información.

Ventajas:

1. Utiliza un token (paquete de información que viaja a través del anillo).
2. Las conexiones punto a punto a lo largo del anillo no sólo proveen un foco fácil de estandarización, sino que también permiten ligar diferentes anillos.
3. La fibra óptica no se adapta bien a las configuraciones de bus, pero puede ser fácilmente acomodada a las topologías de anillo, esto principalmente a que los anillos usualmente transmiten en una dirección.
4. La capacidad de transmisión es compartida equitativamente entre todos los usuarios.
5. Las estaciones que fallan pueden ser aisladas mediante el uso apropiado de ***protocolos***, estos protocolos también proveen adiciones y bajas lógicas de estaciones sin que caiga la velocidad de la red.
6. El ***ruteo*** es sumamente simple.

Un protocolo es un conjunto de normas y regulaciones para la transmisión de datos

El ruteo es la forma en que se direccionan los paquetes de información.

Desventajas:

1. Es difícil aumentar nuevos nodos sin suspender la operación del anillo.
2. Usualmente se necesita un equipo de monitoreo.

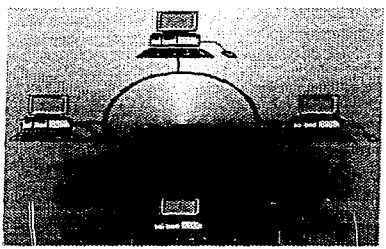


Fig. Topología Anillo

- **Topología Bus**

Los nodos van conectados a una línea totalmente abierta, de tal manera que se puede ir ampliando por simple conexión de nodos en sus extremos .

El término bus se refiere a la línea de transmisión común a todos los nodos. La información transmitida por cualquier nodo viaja a través del bus con el identificativo del nodo al cual va dirigida, y el proceso de recogida de la información se efectuará del mismo modo que en las redes de anillo. Si el nodo reconoce el identificativo y coincide con el suyo, recoge el mensaje, en caso contrario el mensaje sigue recorriendo el bus.

Los equipos conectados al bus, no toman decisiones sobre ruteo o direccionamiento, la responsabilidad de la administración de la red recae en cada nodo a través del protocolo de comunicaciones empleado.

Una particularidad de este tipo de configuración es que una estación no ofrece recursos a las demás, sino que usa los propios.

El bus no tiene controlador central pero cuenta con dispositivos transreceptores en cada punto de conexión con el bus adaptando una configuración que se le denomina ***multipunto***. Debe destacarse que estos transceptores no actúan como generadores de una señal, pero a medida que el mensaje recorre el medio alejándose de la estación transmisora se produce una degradación de las señales eléctricas y en consecuencia habrá una longitud máxima admisible para el medio de transmisión.

Multipunto es una línea que interconecta varias computadoras.

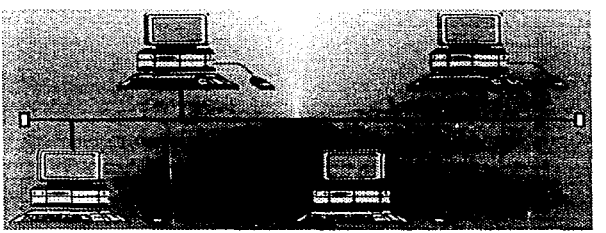


Fig. Topología Bus

Ethernet

Ethernet es una tecnología de transmisión de datos de alta velocidad inventada por Robert M. Metcalfe en 1973, está soporta una topología de bus y usa un canal compartido de comunicaciones, manejado por acceso múltiple por censado de portadora con detección de colisión (CSMA/CD). CSMA/CD no tiene manejo de control central de acceso al canal. Si una estación deseara transmitir a otro lado, tiene que manejar hasta que pueda adquirir el canal. Una vez adquirido este canal, la estación transmite por éste.

Cuando una computadora desea enviar información a otra máquina, forma un paquete con la información, la dirección del destinatario y su propia dirección. Estos paquetes de información son conocidos como "tramas" las cuales son enviadas bit por bit a través del cable coaxial. Las señales en el cable coaxial son bidireccionales (en dos sentidos), de tal manera que todos los sitios de la red detectan el paquete de información. El dispositivo que reconoce la dirección destino como su propia dirección sabe que la trama contiene información destinada a él y por lo tanto tomará la trama y los demás la ignoran.

La implantación Ethernet está referida como la norma 802.3 por el Instituto of Electrical and Electronic Engeneers (IEEE) y es llamada IEEE 802.3.

Redes de Área Amplia (WAN: Wide Area Network)

Una red de área amplia es una red en la que se conectan varias redes locales mediante dispositivos que permiten su conectividad local o remotamente, a pesar de que tengan diferente topología. Estos dispositivos pueden usar o no líneas telefónicas o servicios públicos de transmisión de datos.

Los dispositivos que se usan para formar redes de área amplia son: puentes, ruteadores y gateways.

Los puentes, ruteadores y gateways son "cajas negras" que nos permiten utilizar diferentes topologías y protocolos dentro de un solo sistema heterogéneo. Cada uno de estos elementos tiene ventajas y desventajas, así como aplicaciones específicas.

Con los puentes pueden interconectarse segmentos de red a través de medios físicos diferentes, por ejemplo, es común ver puentes entre cable coaxial y fibra óptica. Además pueden aceptar diferentes protocolos de bajo nivel. Así, en las circunstancias adecuadas, se pueden usar puentes para conectar segmentos similares, como son dos Ethernet, o mezclar segmentos diferentes, como es un Token Ring y uno Ethernet.

TCP/IP es el protocolo de comunicaciones usado en Internet.

Permiten que se comuniquen los dispositivos y los segmentos que usan el mismo protocolo de alto nivel (por ejemplo ***TCP/IP*** o ***IPX***), sin importar cual sea el protocolo de bajo nivel o el estándar de capa física que estén usando.

IPX es un protocolo de comunicaciones de la compañía Novell.

Los puentes son inteligentes, aprenden las direcciones destino del tráfico que pasa por ellos y lo dirigen hacia él. Esto explica su importancia en la división de red: cuando un segmento físico de red tiene tráfico en exceso y su rendimiento está comenzando a degradarse, se le puede dividir en dos segmentos físicos con un puente. Éste dirige el tráfico a su destino final y limita el paso por un determinado segmento. Los puentes usan un proceso de aprendizaje, filtrado y envío para mantener el tráfico dentro del segmento físico al que pertenece.

Debido a que los puentes aprenden direcciones, examinan paquetes y toman decisiones de envío, con frecuencia, su funcionamiento se degrada conforme el tráfico aumenta.

El ruteador en algunos aspectos es más inteligente que el puente. Los ruteadores no tienen la misma capacidad de aprendizaje que los puentes, pero toman decisiones de enrutamiento que determinen la trayectoria más eficiente de datos entre dos segmentos de red.

Los puentes saben cuál es el destino final de la red; los ruteadores

sólo saben dónde se encuentra el siguiente ruteador.

Los puentes toman la decisión de seguir hacia adelante o de eliminar datos en cada paquete dependiendo de si éste está destinado a una dirección al otro lado del puente o no. Los ruteadores eligen el mejor camino para el paquete tras revisar una tabla de enrutamiento. Lo único que consideran son los paquetes dirigidos a ellos por el ruteador anterior o por la estación final de la red, mientras que los puentes deben examinar todos los paquetes que pasan por la red.

Un mainframe es una computadora que da servicios a otras.

Los gateways ofrecen el mejor método para conectar segmentos de red y redes a ***mainframes*** . Se selecciona cuando se tienen que interconectar sistemas que se construyeron totalmente con base en diferentes arquitecturas de comunicación. Por ejemplo se utilizaría un gateway para interconectar una computadora que "hable" TCP/IP con una que hable SNA (System Network Architecture). Las dos arquitecturas no tienen nada en común, por lo que el gateway debe traducir todos los datos que pasan entre los sistemas.

En cada extremo de la red, el gateway ofrece la conversión del protocolo a los segmentos de red conectados en el otro lado. Los gateways no proporcionan enrutamiento de paquetes dentro de los segmentos de red; simplemente entregan sus paquetes de datos de tal forma que los segmentos puedan leerlos. Cuando reciben paquetes del segmento, los traducen y enrutan al gateway en el otro extremo, donde los paquetes vuelven a traducirse y entregarse al segmento de red el extremo opuesto.

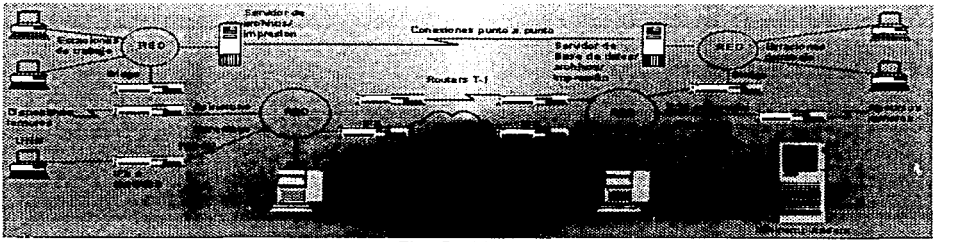


Fig. Red WAN

Redes de Área Metropolitana (MAN: Metropolitan Area Network)

Entre las redes LAN's y las WAN's están las redes MAN (Metropolitan Area Network). Una red MAN es una red en una ciudad completa, pero usa tecnología LAN.

Son una colección de redes de área local, son universalmente aplicables con un alto funcionamiento, son redes de alta velocidad, poseen las mismas características de una red LAN.

Las redes MAN transmiten voz y datos mediante sistemas digitales, su infraestructura puede soportar grandes cantidades de redes locales de cómputo.

Las MAN son de interés en el área digital y en áreas en donde las computadoras están conectadas siempre, a pesar de que ellas pueden usar cable coaxial de banda ancha su transmisión es media.

Los enlaces se pueden realizar vía microondas, fibra óptica y vía satelital.

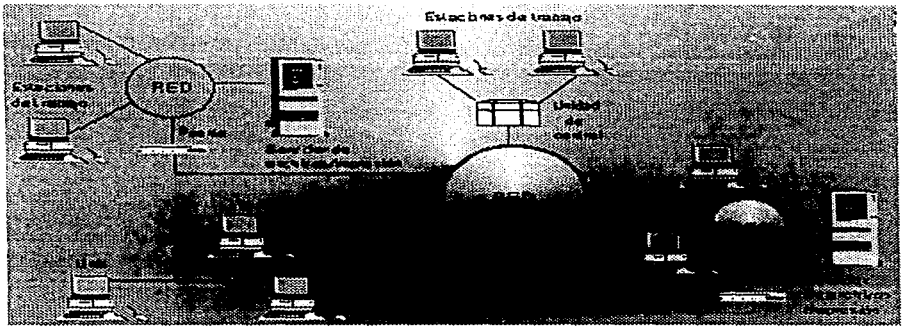


Fig. Red MAN

Modelo OSI/ISO

La International Standarization Organization (OSI) propuso un modelo general de los diferentes niveles en los que opera una red Open System Interconection (OSI).

OSI pone atención al intercambio de información entre sistemas y al funcionamiento interno de cada sistema en particular. Es decir, el Modelo de Referencia OSI constituye el marco de trabajo para el desarrollo de protocolos estándares de comunicación entre dos capas homónimas ubicadas en equipos separados.

El objetivo a largo plazo de OSI es desarrollar una compatibilidad inter-sistemas entre muchos productos y servicios alrededor del mundo.

Este modelo es estratificado y se estructura en siete niveles, como se muestra a continuación:

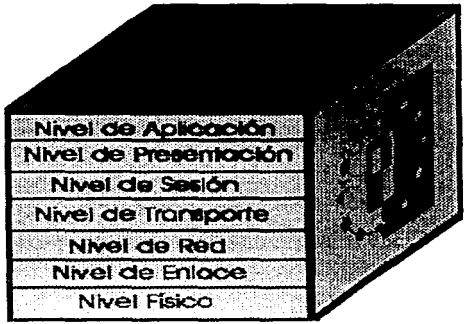


Fig. Modelo ISO/OSI

Los niveles del 1 al 4 se encargan de la transmisión de datos y los niveles del 5 al 7 del uso de éstos, y es en el nivel de transporte en especial donde se llevan a cabo las traducciones de los datos.

Nivel 1: Físico

Provee las características mecánicas, eléctricas, funcionales y de procedimiento, necesarias para establecer, mantener y liberar conexiones físicas entre elementos individuales de equipo. A este nivel se llevan acabo el intercambio de señales eléctricas que representan los datos y la información de control.

- **Nivel 2: Enlace de Datos**

Se encarga de corregir la información, verifica que llegue bien entre nodos adyacentes. Se realiza el reconocimiento de la recepción de datos, así como el control de errores, con la posibilidad de retransmisión en caso de ser necesario.

- **Nivel 3: Red**

Son programas que se encargan de seguir caminos o nodos, tomando bloques de datos del tamaño de paquetes al nivel de transporte y les añade información de dirección y encaminamiento que completan al paquete.

- **Nivel 4: Transporte**

Se encarga de seleccionar el nodo principal y el nodo final; así, éstos pueden ser varios caminos o rutas, es decir proporciona un servicio de transmisión y recepción de datos fiable al nivel de sesión.

- **Nivel 5: Sesión**

Establece y termina una conexión con el proceso en una computadora remota. Debe dar servicio fiable al nivel de presentación y tener la capacidad de restablecer una conexión en caso de que falle uno de los niveles más bajos de la jerarquía.

- **Nivel 6: Presentación**

En él se realiza el formato de los datos, los cuales pueden incluir compresión, traducción y cifrado de éstos

- **Nivel 7: Aplicación**
- Involucra todo en relación con el operador humano.**Nivel 4: Transporte**

Se encarga de seleccionar el nodo principal y el nodo final; así, éstos pueden ser varios caminos o rutas, es decir proporciona un servicio de transmisión y recepción de datos fiable al nivel de sesión.

- **Nivel 5: Sesión**

Establece y termina una conexión con el proceso en una computadora remota. Debe dar servicio fiable al nivel de presentación y tener la capacidad de restablecer una conexión en caso de que falle uno de los niveles más bajos de la jerarquía.

- **Nivel 6: Presentación**

En él se realiza el formato de los datos, los cuales pueden incluir compresión, traducción y cifrado de éstos

- **Nivel 7: Aplicación**

Involucra todo en relación con el operador humano.

Red UNAM

La Universidad Nacional Autónoma de México definió un ambicioso programa institucional que respondiera a la necesidad de modernizar las comunicaciones en un plazo relativamente corto; a finales de 1989 se estableció un ambicioso proyecto, que debía quedar terminado en 36 meses, con la capacidad de crecer conforme a las necesidades de la institución. En particular, el sistema debería contemplar la transmisión indistinta de voz y datos y completar a las redes de computadoras de reciente instalación. A este sistema se le conoce como red integral de telecomunicaciones de la Universidad Nacional Autónoma de México, los objetivos que se persiguen con esta gran red de redes son:

- Promover el intercambio de ideas, pensamientos y opiniones que enriquezcan a los pueblos, instituciones e individuos.
- Apoyar el crecimiento de la UNAM y de México, proporcionando una opción para el tránsito libre de información entre las diversas instituciones generadoras y transformadoras de conocimientos del país y del mundo.
- Permitir que otras fuentes de conocimiento tengan acceso a los bancos de información .

La estructura principal de RED UNAM es un anillo de ***FDDI***, que enlaza 5 enrutadores principales. Conectadas a ellos se encuentran las redes locales de cada dependencia, las que se encuentran en el campus de C.U. son desplazadas por fibra óptica.

FDDI (Fiber Optic Data Interface) es una interfaz de distribución de datos de fibra óptica y trasmite a 100 megabits por segundo.

Aquellas que se hallan fuera de él, se comunican con RED UNAM a través de alguno de los siguientes medios:

1. Dentro del área metropolitana

- Radio módem
- Líneas conmutadas o privadas
- Microondas
- ***RDI***

2. Resto de la República Mexicana
   - RDI
   - Enlaces satelitales

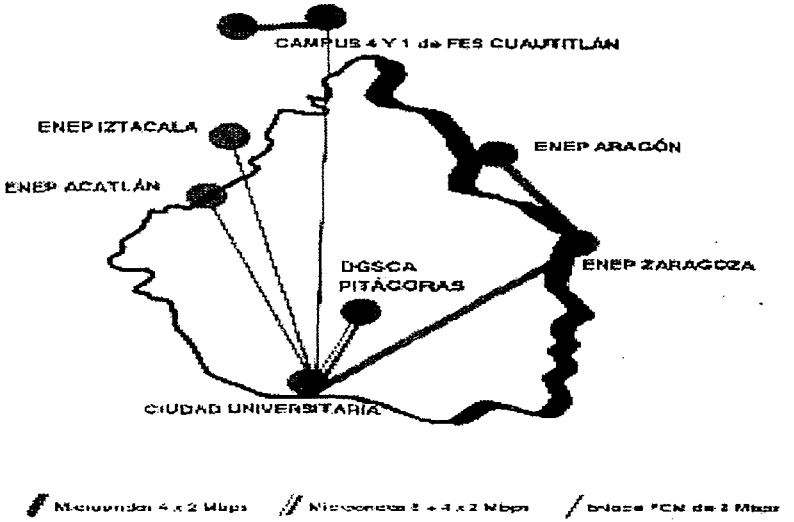


Fig. Enlaces en REDUNAM

En las redes de la UNAM las topologías mas empleadas son variantes de Ethernet: en primer lugar se tienen las redes tipo estrella, es posible encontrar este tipo de redes complementado con verticales de coaxial grueso en edificios de varios pisos. El segundo medio mas empleado es el coaxial delgado, aunque su uso empieza a decaer debido a sus desventajas frente al par trenzado.

La RED UNAM utiliza el protocolo de comunicación TCP/IP, él cual permite la comunicación en Internet y la instalación de sistemas operativos de red como Windows NT y sus variantes, LAN Manager y NetWare.

Host es una computadora central o controladora e un entorno de procesamiento distribuido.

Dentro de los servicios que brinda RED UNAM están aquellos que permiten la comunicación entre los diversos ***hosts*** (máquinas capaces de dar servicio de red), ruteo sin el cual no sería posible la comunicación con los hosts de otras redes, DNS (Domain Name Service) que resuelve la conversión entre las direcciones de las máquinas, de forma que el usuario pueda utilizar el nombre tlaloc.dgapa.unam.mx para comunicarse con su servidor de correo en lugar de la dirección 132.248.37.2, lo cual a su vez esta ligada a una dirección física de la tarjeta de red que permite su identificación única, esta dirección se conoce como hardware address y es fijada de fábrica NOC/NIC (Network Operation Center/ Network Information Center) que se encargan de atender el monitoreo y mantenimiento de la red, así como de las asesorías requeridas por los usuarios.

Con base en estos servicios, los usuarios pueden hacer uso de otros, como son: Servidores de correo, Gophers, Archie y Servicios de Telnet.

Internet

Internet es una red mundial de computadoras. Se calcula que integra a más de 80 millones de usuarios. Pero, sin duda, su gran atractivo es la enorme cantidad de servicios que ofrece.

Al inicio de los 60's se desarrollaron las primeras "redes de conmutación de paquetes". En este tipo de redes la información que se envía se subdivide en pequeñas partes o paquetes, que son dirigidos hacia el receptor (a veces por diferentes caminos). Una vez allí, todas las partes se unen en el orden correcto, para recuperar la información original.

Con esta tecnología se aseguraba que varios usuarios podrían mandar mensajes por las mismas líneas de comunicación y, lo que es más

importante, no se establecerá ninguna dependencia de un denominado host central.

El backbone e la parte de una red que soport el mayor tráfico (es la "espina dorsal").

Basándose en todas estas investigaciones, el ministerio de defensa de los EUA para 1978 se tenían definidos los principales protocolos y la arquitectura los cuales se emplearon en la red experimental llamada ARPANet (Advanced Research Proyect Agency NETwork; Red de la Agencia de Investigación de Proyectos Avanzados). Al conjunto de protocolos definidos se les llamó TCP/IP en honor a los principales protocolos (TCP e IP) de esta tecnología. El proyecto ARPANet tenia una finalidad: comunicar todos los centros militares. Muy pronto ARPANet fue el ***backbone*** sobre el cual se unieron mas redes y a la nueva red resultante se le llamó Internet.

Se tomó como estratégia de difusión de esta tecnología el poner a disposición de las Universidades una implementación a bajo costo, además de que se realizó una implementación de TCP/IP para su uso con el sistema operativo UNIX desarrollado en la Universidad de Berkeley.

Se formó el INWG (InterNetwork Working Group) como foro de discusión para crear las bases de una gran red mundial: Internet. A partir de entonces se han realizado una gran cantidad de foros de investigación mundiales para crear protocolos y sistemas estándares para la interconexión de host en dicha red. En 1983 ARPANet se dividió en ARPANet y MILNet.

La última es la que formó la red de defensa EE UU. ARPANet quedó como un anillo central que unía redes más pequeñas y centros de supercomputación de la NSF (National Scientific Foundation). Finalmente, esta última creó su propio anillo llamado NSFNet, que acabó con la ARPANet en 1990.

Desde el punto de vista técnico, Internet es un gran conjunto de redes de ordenadores interconectadas. Desde otro punto de vista, Internet es un fenómeno sociocultural. Un usuario desde su consola, tiene acceso a la mayor fuente de información que existe.

Los principales servicios que Internet ofrece a los usuarios son los siguientes:

mail, telnet, FTP, FTP anonymous y archie, los cuales mencionamos a continuación.

Mail: Correo electrónico

El Correo Electrónico es el servicio más básico y el más antiguo. El correo electrónico permite intercambio de mensajes entre distintos usuarios.

El uso de correo electrónico está cambiando las formas de trabajo que tienen los usuarios de Internet. Hay muchas organizaciones que, aún trabajando en proyectos científicos y de investigación muy importantes, tienen a sus especialistas distribuidos por todo el mundo, intercambiando sus avances a través del correo electrónico, entre otros medios.

Como en cualquier sistema postal, una persona envía una carta a otra especificando su nombre, dirección, código postal, etc. Así, en Internet, para enviar correo se debe concretar la dirección del destinatario, expresada de la siguiente forma: <nombre_usuario>@<direccion_host>. La dirección de la computadora receptora puede ser expresada de las dos formas que se contemplan en Internet, por su dirección numérica, o bien, con el nombre que corresponde a esa dirección.

Ejemplo (envío de Correo Electrónico):

Si se le quisiera enviar un correo al director de la DGAPA, escribiríamos lo siguiente:

```
%mail boldu@tlaloc.dgapa.unam.mx
Subject: Un mensaje de prueba
Esto es sólo un mensaje de prueba... disculpe las molestias, Doctor ... :-)
.
```

En donde el símbolo '%' representa el "prompt" o indicador del sistema operativo UNIX. Mail es el comando para enviar/revisar el correo y el "Subject" es simplemente un encabezado que llevan los mensajes para indicar de alguna forma su contenido. Como se puede ver al final del ejemplo

anterior, para finalizar el mensaje se escribe un punto (".") y se pulsa <Enter>.

La próxima vez que el doctor desee leer el correo, lo hará de la siguiente forma:

Ejemplo (lectura de Correo Electrónico):

```
%mail
From plantac Sun Oct 13 11:22 CST 1996
Date: Sun, 13 Oct 1996 11:22:55 +0600
Subject: Un mensaje de prueba
Content-Length: 76

Esto es solo un mensaje de prueba... disculpe las molestias, Doctor... :-)
?_
```

En este caso se observa cómo también se ejecuta el comando "mail" para leer el correo. Al entrar en esté se indica quién envío el correo (plantac), la fecha de envío (Sun, 13 Oct 1996 11:22:55 +0600), el encabezado (Un mensaje de prueba), la longitud del mensaje en caracteres (76) y el cuerpo del mensaje (Esto es solo un mensaje de prueba... disculpe las molestias, Doctor... :-)). El símbolo "?" es el indicador de comandos de mail, en el se puede teclear el número de correo que se quiera ver, por ejemplo 2, o bien "n" para ver consecutivamente los correos existentes, "d #" para borrar el correo número #, "q", "x" ó "Ctrl+D" para salir, "s" para salvar los correos a un archivo y "?" para ver ayuda sobre algunos otros comandos de mail.

Telnet: Conexión remota

Telnet (Terminal Network) es un programa de aplicación que permite la conexión entre computadoras dentro de la Internet. Permite la conexión remota de un usuario a otra computadora (incluso si éste se encuentra a miles
de kilómetros) como si se tratase de una terminal directamente conectada a dicha computadora.

Mientras que las conexiones por módem están limitadas por la calidad de las líneas telefónicas, el servicio de Telnet proporciona una conexión libre de errores.

Al igual que en el caso de otros servicios de Internet, el comando actual para negociar una conexión a través de Telnet depende mucho del sistema operativo en el cual se encuentra el usuario. El más usual (tanto en sistemas operativos UNIX como VMS) es Telnet, y tiene la siguiente forma:

telnet <computadora_destino> [puerto]

Donde la *<computadora_destino>* puede ser expresada por su dirección IP o bien por su nombre completo y, el <puerto> es un parámetro opcional, que es necesario incluir en el caso de acceder a determinados servicios.

Ejemplo de una sesión con Telnet:

```
$telnet tlaloc.dgapa.unam.mx
Trying 132.248.37.2 ...
Connected to tlaloc.dgapa.unam.mx.
Escape character is '^]'.

UNIX(r) System V Release 4.0 (tlaloc)

        Bienvenido a Tlaloc
login: yojuana
Password:********
Last login: Wed Jul  3 19:15:16 from 132.248.37.5
Sun Microsystems Inc.   SunOS 5.4       Generic July 1994
You have new mail.
tlaloc 1:>
```

El significado de esta sesión es el siguiente:

"$" es el prompt del shell de UNIX, e indica que se espera un comando para ejecución. "telnet" es el comando de conexión, y "tlaloc.dgapa.unam.mx" es aquella computadora a la que se desea conectar.

FTP: Transferencia de archivos

FTP (File Transfer Protocol: Protocolo de Transferencia de Archivos) permite transferir archivos sobre Internet entre una máquina local y otra remota.

Los comandos básicos de FTP son:

open <dirección IP>	Abre una sesión FTP en la computadora indicada.
dir	Lista de archivos del directorio de la computadora a la que nos hemos conectado.
pwd	Visualiza el directorio remoto en el que estamos situados.
dir	Lista el contenido de directorio actual.
cd <directorio>	Cambio al directorio especificado.
lcd <directorio>	Comando de movimiento para directorios locales.
binary	Establece modo binario de transferencia.
ascii	Establece modo ascii de transferencia. Sólo para archivos de texto.
get <directorio>	Copia un archivo desde la computadora remota.
put <dierctorio>	Transfiere un archivo a la computadora remota.
bye	Termina la sesión FTP.

FTP Anonymous: Transferencia de archivos anónima

Los servidores FTP anonymous son grandes "cajones" de archivos distribuidos y organizados en directorios. Contienen programas (normalmente de dominio público), archivos de imágenes, sonido y video.

El medio de acceso y recuperación de la información es FTP (File Transfer Protocol). Para entrar en estos servidores, tecleamos FTP y nombre del servidor. El sistema nos pregunta login, a lo que respondemos con la palabra "anonymous" y en el password le indicaremos nuestra dirección de correo electrónico. Algunos servidores autentifican esta dirección, con lo que restringen el acceso a ciertos usuarios.

Al existir miles de servidores FTP anonymous, se hace imprescindible una herramienta de búsqueda. Archie es una solución implementada para este fin.

Ejemplo de una sesión con FTP anonymous:

```
$ftp bigblue.pvv.unit.no
Connected to bigblue.pvv.unit.no.
220 bigblue.pvv.ntnu.no FTP server (Version wu-2.4(4) Thu Feb 9 16:47:32 NFT
199
5) ready.
Name (bigblue.pvv.unit.no:papiit): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: yojuana@tlaloc.dgapa.unam.mx
230-Please read the file README
230-  it was last modified on Fri Apr 28 22:55:18 1995 - 440 days ago
230 Guest login ok, access restrictions apply.
ftp>
```

Archie: Búsqueda de archivos en la Internet

El sistema de búsqueda de archivos a través de los servidores FTP anonymous llamado Archie está considerado como un servicio independiente del Telnet, aunque casi siempre se utiliza este último para acceder a él.

Ejemplo de una sesión con Archie:

```
$telnet archie.funet.fi
Trying 128.214.248.46 ...
Connected to archie.funet.fi.
Escape character is '^]'.

SunOS UNIX (orava)

login: archie
# Bunyip Information Systems, Inc., 1993, 1994, 1995

# Terminal type set to `vt100 24 80'.
# `erase' character is `^?'.
# `search' (type string) has the value `sub'.
FUNET-archie> prog UNAM
# Search type: sub.
# Your queue position: 1
# Estimated time for completion: 5 seconds.
working... =   0
Host bigblue.pvv.unit.no    (129.241.210.232)
Last updated 22:32 20 Apr 1996
Location: /store/store/ernie/xemacs/src-19.13-local/lisp/packages
FILE    -rwxr-xr-x      43 bytes  05:00  6 Sep 1995  netunam.el
    FILE    -rwxr-xr-x      44 bytes  05:00  6 Sep 1995  netunam.elc
:q
FUNET-archie>exit
$_
```

Como se puede ver primero se accede al host que da el servicio Archie, en este caso "archie.funet.fi". Una vez conectado el servicio pide un nombre de usuario, se debe introducir "archie" cuando pregunte por el "login".

Cuando aparezca el indicativo de Archie, "FUNET-archie>", se introduce "prog <texto a buscar>". Al poco tiempo el servidor Archie dará una lista de servidores FTP anonymous, así como los correspondientes directorios donde se encuentre el archivo que contenga el texto a buscar. Para terminar con la búsqueda se debe teclear "q" desde el indicativo de búsqueda ":". Finalmente, para terminar la sesión basta teclear "exit" en el indicativo de Archie.

Importante, se debe de observar el nombre del Host y el directorio que contiene el archivo encontrado, para después realizar una transferencia de archivos (FTP).

Algunos servidores Archie son:

- sun.redirios.es
- archie.funet.fi
- archie.leth.se

Es importante hacer notar que desde el punto del programador de aplicaciones, la Internet provee de dos tipos de servicios sobre los cuales trabajan las aplicaciones:

Conecctionless Packet Delivery Service: Es una forma de envío de datos a través de la red, en la cual TCP/IP rutea el mensaje de una maquina a otra con base en la información de direccionamiento incluida en el mensaje. Al enviar este servicio, cada parte del mensaje de manera independiente, no garantiza que llegue en un orden adecuado. Trabaja de una forma mas estrecha con el hardware, lo que lo hace muy eficiente.

- Reliable Stream Transport Service: Aunque en el servicio anterior se muestra gran eficiencia, muchas aplicaciones requieren mas que un servicio de entrega de mensajes. Necesitan de una conexión confiable con métodos de recuperación de errores, capacidad de retransmisión, etc. Este tipo de servicio cubre con estas necesidades. Permite que una aplicación establezca "una conexión" con la aplicación en el otro equipo, y enviar un gran volumen de datos a través de ella; como si fuera un enlace dedicado entre ambas.

Estas características son comunes a distintas redes de protocolos, sin embargo las caracteristicas, que distinguen a TCP/IP son:

- Independencia de la tecnología de red sobre la que trabaje
- Interconexión universal, a cada computadora se le asigna una dirección única y que la identifica a través de todo la red .
- End-to-end acknowledgments, los protocolos de comunicación proveen un sistema de acknowledgments (acuse de recibo) entre máquina origen y la destino, en lugar de que esto ocurra entre las diferentes máquinas a las que pueda llegar el paquete de información durante su viaje.
- Protocolos estándar para las aplicaciones de usuario como lo son la transparencia de archivos y el correo electrónico.

❖ World Wide Web

El World Wide Web también llamado WWW, W3 ó Web es un servicio de Internet; en general, representa a todas las computadoras (servidores) que ofrecen a los usuarios acceso a la documentación e información basada en hipertexto. La Web elimina el uso de la línea de los sistemas operativos para lograr transferir un archivo o conectarse a otra computadora. Ofrece recursos que incluyen imágenes, gráficas, fotografías, audio y video, que

permiten al usuario interactuar de forma más práctica y divertida con la información en Internet.

La Web hace uso de visualizadores para poder acceder a documentos dentro de Internet. A partir de 1993 el uso de visualizadores revolucionan el uso de Internet. En los primeros meses existían unas 50 diferentes páginas que podían ser desplegadas. Para mediados de 1993 ya se contaba con unas 1,500 páginas y hoy en día existen millones de lugares o centros de información que pueden ser visitados.

Arquitectura Cliente/Servidor

La arquitectura cliente/servidor es una forma de cómputo en red en el que ciertas funciones solicitadas por clientes son servidas por los procesos adecuados.

En un sistema cliente/servidor, uno o más clientes y uno o más servidores, junto con el sistema operativo y los protocolos de comunicación, conforman el ambiente que permite y facilita el cómputo distribuido.

En una aplicación basada en esta arquitectura existen dos procesos independientes, en lugar de uno solo. De esta forma se puede repartir el trabajo a través de varias computadoras en una red. Estos dos procesos, cliente y servidor, se comunican mediante un protocolo bien definido. Esta técnica permite la comunicación entre distintas computadoras (servidores de archivos, estaciones de trabajo con alta calidad de graficación, etc.), para que cada una de ellas se dedique a realizar el trabajo que hace mejor.

De manera introductoria, se puede decir que un servidor es un sistema o un programa que provee de algún servicio a otros sistemas a través de una red. Un ejemplo típico es un servidor de archivos, el cual permite el acceso a información remota a cualquier usuario a través de la red. Un cliente es un sistema que requiere y recibe alguna acción de un servidor.

Un servidor es un administrador de recursos. Un recurso puede ser identificado como algo físico (por ejemplo una impresora) y un back end

(por ejemplo un servidor). En el estándar de la ciencia de la computación, un proceso es un programa corriendo en un procesador.

De manera general, para que se inicie la comunicación entre un cliente y servidor es necesario establecer una sesión. Por lo tanto, el servidor debe estar esperando ("escuchando") que algún cliente trate de establecer una sesión. Esto quiere decir que un cliente puede "hablar" pero si no es "escuchado", la comunicación va a fracasar. Es muy posible que, por algún momento el servidor también "hable" y que el cliente "escuche", pero esto sólo se hará cuando el servidor así se lo indique al cliente.

Un servidor también se reserva el derecho de establecer comunicación con uno o más clientes. Así, el servidor se encargará de atender a cada cliente y establecer los mecanismos que seguirán para la distribución de sus servicios. Un servidor define operaciones que son exportadas a los clientes. Los clientes invocan estas operaciones para que el servidor controle el manejo de datos.

Típicamente, una aplicación (cliente) comenzará una transacción (mediante una sesión), ejecutará una o varias operaciones en el servidor y terminará la transacción (terminando la sesión). Lógicamente, los servidores están estructurados como un ciclo infinito. El servidor simplemente recibe los requerimientos de los clientes para invocar operaciones en favor de esas transacciones. Para implantar las operaciones que exporta, el servidor puede requerir de otro servicio o puede manipular sus propios datos.

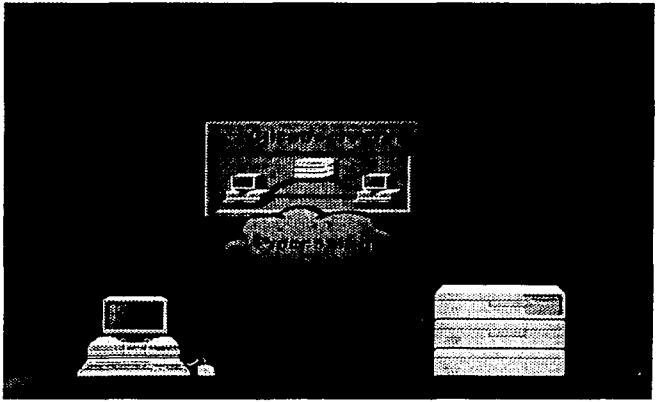


Fig. Arquitectura cliente/servidor: "ciertas funciones solicitadas por clientes son servidas por los procesos adecuados".

Una aplicación de bases de datos que usa arquitectura cliente/servidor requiere servicios de una base de datos de un servidor a través de sentencias ***SQL***. La base de datos puede estar localizada virtualmente en la máquina del cliente o en otra maquina. Los mensajes son pasados entre el cliente y el servidor. Una vez que los datos requeridos han sido pasados al cliente, el procesamiento toma lugar a nivel del cliente. En otras palabras, los datos pueden ser disponibles para muchos usuarios, pero el procesamiento toma lugar localmente.

En el modelo del cliente las presentaciones y funciones son localizadas y los datos residen en el servidor.

SQL (Structure Query Language) es un leguaje utilizado para procesar datos en una base d datos relacional.

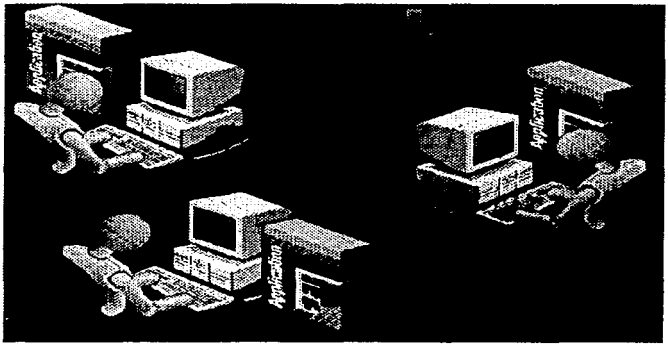
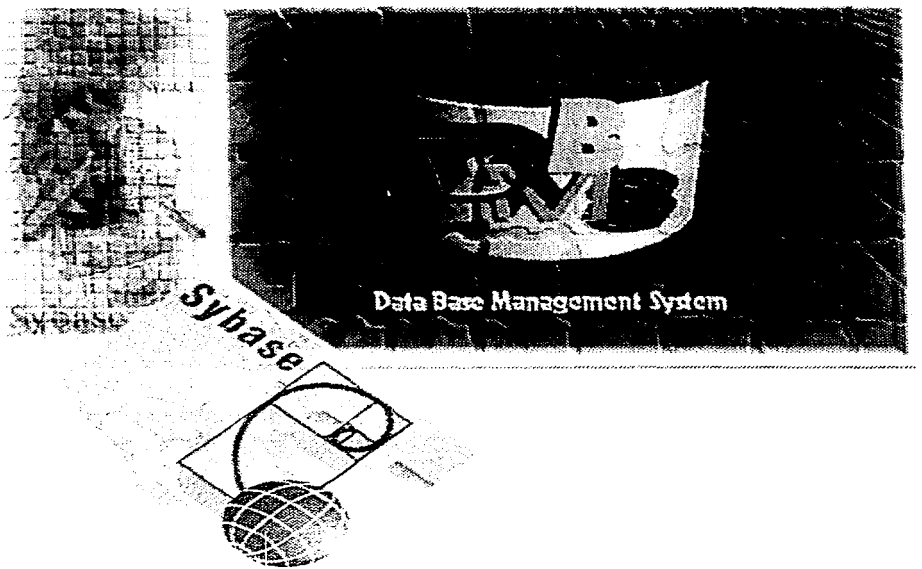


Fig. Aplicación de bases de datos: "las aplicaciones realizan consultas al servidor SQL y procesan los datos para mostrarlos al usuario".

Un caso del esquema cliente/servidor es WWW, aquí los sevidores son programas que utilizan el protocolo Hyper Text Transfer Protocol (***HTTP***) para distribuir la información, el programa cliente establece una conexión con el servidor utilizando el protocolo de comunicaciones TCP/IP. Si este acepta la conexión, el cliente hace una petición de la información en un formato muy simple, con la dirección del documento o archivo requerido. El servidor envía entonces al cliente esta información o archivo en un mensaje, generalmente con formato de hipertexto y una vez realizado la transmisión, el servidor cierra la conexión.

HTTP es un protocolo con l ligereza y velocidad necesaria para distribuir y manejar sistemas de información hipermedia.

4 Bases de Datos

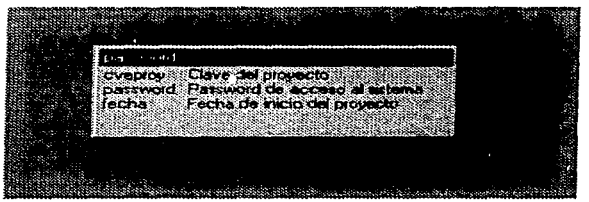


Las bases de datos relaciónales fueron introducidas por el Dr. Codd en 1970 usando un conjunto de teorías matemáticas para describir las relaciones entre los datos. Definió como una base de datos relacional aquella base de datos que el usuario percibe como un conjunto de relaciones normalizadas que varían con el tiempo.

En las bases de datos relaciónales los datos son alineados en conjuntos llamados ***tablas***. Cada tabla consiste de un número de renglones, ,llamados registros, con la misma estructura; esto es, cada renglón tiene el mismo número y tipo de columnas. Las columnas de un registro son también llamadas ***atributos***.

Tabla: Conjunto de renglones (registros)y columnas (atributos) que almacenan información.

Atributo: Información que se necesita conocer o tener de una tabla.



(a)

cveproy	password	fecha
IN205393	COLORDI	93
IN304293	LIRA4	93
IN601393	MEX@234	93
IN101094	SIN%SIN	94
IN101793	KIPER	94
IN210094	MAYNION	94
IN301395	RESITA	95
IN301695	PAZ3R	95
IN501594	ROCTE	95

(b)

Fig. Una tabla consta de atributos llamados columnas (a), los registros en una tabla tienen la misma estructura (b).

Una base de datos relacional esta hecha de un número de tablas. Las ***relaciones*** están definidas por campos que son comunes entre tablas,

Relación: Asociación entre dos o más tablas

es decir, son del mismo tipo y tamaño y hacen referencia a la misma información.

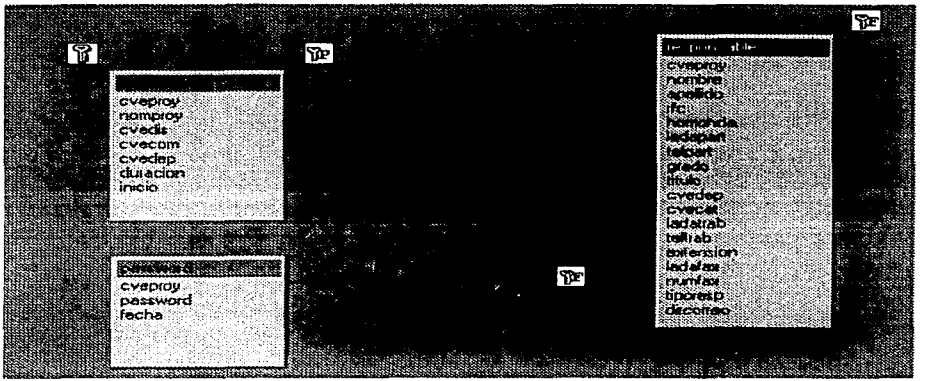


Fig. Una base de datos se compone de varias tablas, las cuales se relacionan por un campo común(cveproy).

El orden de los datos en la tabla no es significativo, tampoco necesita de un orden cuando los registros están incluídos en la relación.

En pocas palabras las bases de datos relaciónales utilizan un modelo para mostrar cómo se relacionan lógicamente los datos de un registro. Cada tabla tiene una columna o combinación de columnas cuyos valores identifican cada registro. A esta columna o columnas se le denomina ***llave primaria*** de la tabla, pues su valor es único para cada registro, de modo que no deben existir dos registros de una tabla con llave primaria que sean duplicados exactos; ni tampoco puede permitir valores nulos. Las llaves primarias pueden ser usadas para hacer un ordenamiento físico de los datos dentro de la tabla.

Llave Primaria: Columna o conjunto de columnas cuyos valores identifican a cada registro.

cveproy
IN205393
IN304293
IN601393
IN101094
IN101793
IN210094
IN301395
IN301695
IN501594

Fig. Una llave primaria identifica los datos de un registro, es única (los valores no se repiten) y no permite valores nulos.

Una columna o combinación de columnas de una tabla cuyo valor coincide con la llave primaria de alguna otra tabla se denomina ***llave foránea***, la cual no es única y puede permitir valores nulos.

Llave Foránea: Columna o columnas cuyos valores coinciden con la llave primaria de alguna tabla.

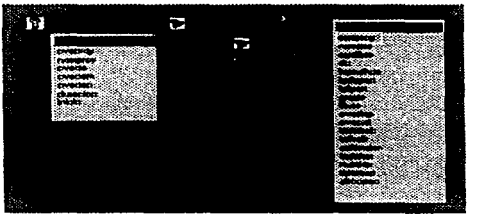


Fig. Una llave foránea no es única y puede permitir valores nulos.

El modelo de bases de datos relaciónales permite una completa flexibilidad en la definición de relaciones de los datos, por lo que las tablas pueden ser creadas o modificadas sin afectar otras tablas. También reducen considerablemente la duplicidad de datos, por lo que puede mantener la integridad de los datos, esto es su validez y consistencia. Finalmente, proporciona un mantenimiento fácil.

Diseño

El diseño de bases de datos en un ambiente relacional es mucho más fácil que en otros modelos de bases de datos porque la base de diseño lógico está separada de la fase de diseño físico.

Uno de los objetivos principales de las bases de datos relaciónales es el de evitar la redundancia entre datos y prevenir la ocurrencia de múltiples copias de un mismo dato, pues un dato podría ser inconsistente si existieran múltiples copias del mismo y no se actualizaran todas estas copias a un tiempo. Lo que caría origen a una base de datos inconsistente que proporcione informacion incorrecta o contradictoria.

Existen básicamente dos escuelas de diseño de Base de Datos Relaciónales:

- Modelo de Entidad/Relación
- Modelo de Normalización

Modelo Entidad/Relación (E/R)

El ***Modelo de E/R*** tiende a ser realizado con la ayuda de diagramas que permite entender el manejo de los datos, el modelo E/R hace una representación simplificada de la realidad, considera las relaciones entre ***entidades***. Las entidades son aspectos importantes acerca de los cuales necesitamos saber o conocer información


Modelo Entidad Relación: Representación simplificada de la realidad considerando relaciones entre las entidades.


Entidad: Aspectos importantes acerca de los cuales necesitamos obtener información.

El modelo E/R considera las relaciones entre entidades. Las entidades son cosas o conceptos acerca de los datos que son recopilados. El diseño genera una ***relación*** entre entidades con los atributos de las columnas.


Relación: Asociación entre dos o más entidades o entre una entidad consigo misma.

El modelo E/R se basa en los modelos siguientes:

1) Modelo de Datos Relacional

Es una colección finita de tablas (de 2 dimensiones) formada por columnas y renglones que representan una situación.

Tiene las siguientes Características

1. Simplicidad

Las tablas son una forma familiar y explicable por sí misma para representar los datos. La mayoría de la gente ha utilizado datos en forma de tabla, no se requiere de un entrenamiento especial para entender y utilizar los datos que se representan en las tablas. En pocas palabras son amigables a los usuarios.

2. Precisión

Las tablas correctamente diseñadas conservan la integridad, dicen lo que significan y significan lo que dicen. Pueden ser implementadas y procesadas con una gran variedad de configuraciones de software y hardware. En pocas palabras son amigables con las computadoras.

3. Flexibilidad

Las tablas no solamente muestran la estructura de los datos sino que pueden mostrar los datos también. Esto nos permite manejar el modelo antes de implementarlo. En pocas palabras las tablas son apropiadas no solo para modelar datos sino para procesarlos también.

2) Modelado de Entidad

Para modelar una entidad se definen los siguientes pasos:

1. Descubrir entidades (Examinar sustantivos)

2. Definir el alcance de la entidad (es importante definir si la entidad es de interés al sistema)

3. Definir un identificador único

4. Documentar

5. Incluir en el diagrama Entidad/Relación

Existen estándares de diagramación de entidades:

- Cajas de cualquier dimensión con las esquinas redondeadas
- Un nombre único para cada entidad
- Nombre de la entidad en mayúsculas y singular
- Nombre de los atributos en minúsculas

3)Modelado de Relaciones

Una relación es bidireccional y representa la asociación entre dos entidades o entre una entidad consigo misma.

Cada dirección de una relación tiene:

- Un nombre: Ej. enseñado o asignado
- Una opción: Ej. Debe ser o puede ser
- Un grado: Ej. Uno y solamente uno

Estándares de diagramación

- Una línea entre dos entidades
- Nombre de relaciones en minúsculas
- Opcionalidad

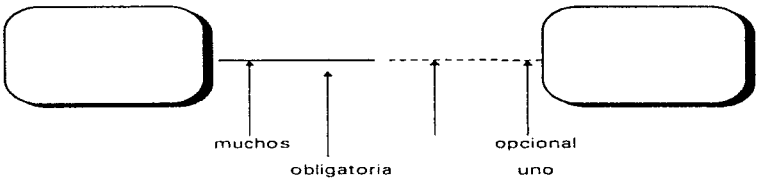
Opcional ----------

Obligatoria ——————

- Grado

Uno o más

Uno y solamente uno



Pasos para el modelado de una relación:

1. Descubrir la relación entre las entidades
2. Definir el alcance de una relación (definir si la relación es importante para el sistema)
3. Definir el tipo de relación

   Las relaciones se catalogan en los siguientes tipos:

   - Una relación muchos a uno (M:1 o M a 1) tiene grado uno o más en una dirección y el grado de uno y solamente uno en la otra dirección.
   - Una relación muchos a muchos (M:M o M a M) tiene el grado de uno o más en ambos direcciones.
   - Una relación uno a uno (1:1 o 1 a 1) tiene un grado de uno y solamente uno en ambos direcciones.

4. Documentar las entidades en un diagrama Entidad/Relación

4) Modelado de Atributos

- Los atributos son información que se necesita conocer o tener acerca de la entidad. Los atributos describen una entidad para calificar, identificar, clasificar, cuantificar o expresar el estado de una entidad.

Estándares de diagramación

- Los nombre de los atributos están en singular y se muestran en minúsculas
- Listar los nombre de los atributos en su caja de entidad

Pasos para el modelado de un atributo:

1. Descubrir atributos
2. Definir el alcance de atributo (Decidir si el atributo es de interés al sistema)
3. Determinar una llave primaria para el atributo (determinar cual es la mejor opción para colocar el atributo en una tabla)
4. Documentar el atributo en forma de tabla

Ventajas del modelo E/R

- Buena cuando se no todos los atributos son conocidos
- Buena para bases de datos muy complejas o largas (tiene muchas relaciones)
- Los diagramas son fáciles de leer
- Más orientada al sentido común que a las matemáticas
- Buena para el análisis posterior de bases de datos y consultas que corran en una base de datcs
- Las dependencias de múltiples valores son fácil de entender
- Un buen medio para propósitos de discusión

Desventajas

- El diseñador evalúa el grado de normalización (Depende del criterio del diseñador)
- Se debe normalizar
- No es fácil definir e identificar entidades

Modelado de Normalización

El modelado de Normalización trabaja a un nivel bajo y distribuido entre los campos directamente. Una lista de dependencias funcionales son construidas y descompuestas (normalizadas) en conjuntos de dependencias que conducen a las relaciones. Esta descomposición a través de etapas que conducen a las relaciones, es conocida como *Forma Normal*.

Las formas normales comúnmente usadas son las siguientes:

Primera forma Normal (1FN)

Dada la tabla T, con una llave primaria P y un atributo A, se dice que T está en primera forma normal, si y solo si el valor del atributo A en cualquier renglón depende del valor de la llave primaria P en ese renglón. Es decir la tabla debe tener un solo valor por cada renglón. La tabla no puede contener grupos repetitivos.

Segunda Forma Normal (2FN)

Dada una tabla T en primera forma normal con una llave primaria P en varias columnas, con componentes P_1,P_2 y un atributo A, se dice que T está en segunda forma normal, si y solo si el valor del atributo A en cualquier renglón depende de los dos valores P_1 y P_2 en ese renglón. La tabla debe estar en 1FN , cada columna que no es llave debe ser dependiente de la llave primaria completa.

Tercer Forma Normal

Dada una tabla T en segunda forma normal, con llave primaria P y dos atributos A_1 y A_2, se dice que T esta en tercera forma normal si y sólo si el valor del atributo A_1 en cualquier renglón no depende del valor del atributo A_2, este marcado con ND_1 y el valor del atributo A_2 no dependa del valor del

atributo A_1 en ningún renglón a menos de que A_1 este marcado con ND. La tabla debe estar en 2FN. Una columna que no es llave primaria no debe depender de otra columna no llave.

¿ Por que Normalizar?

La normalización minimiza la redundancia de los datos; un dato sin normalizar es redundante

La redundancia de datos causa problemas de integridad. Las transacciones de actualización y borrado puede no ser consistente en todas las copias de los datos causando inconsistencia en la base de datos.

La normalización ayuda a identificar entidades, relaciones y tablas mal diseñadas.

Sybase

Es el pionero en la evolución de mainframes y ***arquitectura cliente/servidor*** con la introducción de Sybase SQL Server en 1987, el primer sistema manejador de bases de datos relaciónales (***RDBMS***) designado específicamente para un medio ambiente de sistemas abiertos.

Sybase soporta demandas complejas que requieren un proceso intenso de transacciones en línea (OLTP), velocidad y confiabilidad. Su adaptabilidad proporciona a organizaciones del World Wide Web una sustanciable ventaja, haciendo a ellas más competitivas.

Sybase System 10 es la sucesor de Sybase 4.x, Sybase System 10 tiene improvisaciones como son:

- Habilidad para alojar más memoria en el ***servidor*** sin degradar la eficiencia
- Habilidad para atender más de una base de datos y tomar ventaja de multiproceso en las máquinas

Arquitectura: Es un conjunto de estrategias que se utilizan para resolver problemas de implantación de sistemas.

Cliente: Es un programa que ofrece una petición a un servidor que responde a una demanda.

Servidor: Dispositivo de hardware o subrutina de software que provee uno o mas servicios predefinidos a una población de entidades usuarias.

Manejador de Bases de Datos (RDBMS): Conjunto de programas que sirven para manipular las tablas que conforman una Base de datos.

- Habilidad en el respaldo de la información

Sybase System 10 se compone de uno o más **servidores *SQL Server*** que corren bajo UNIX, es decir, un SQL Server es un proceso en Unix. Este también es conocido como el motor de la base de datos. Cada servidor puede soportar hasta 255 bases de datos.

Si el servidor SQL es sobrecargado, es factible pensar en crear otro servidor SQL para cubrir todas las necesidades y evitar problemas de carga de trabajo para el servidor. Un servidor SQL soporta conexiones a éste. Una conexión puede ser vista como un usuario, pero no necesariamente, esto es un usuario puede tener múltiples instancias de su aplicación y cada una de éstas tiene su propia conexión al servidor SQL.

Las conexiones toman recursos del servidor SQL, es decir disminuyen la ***memoria cache*** disponible del servidor SQL, el servidor Sybase System 10 tiene disponible 300 MB lo que mejora considerablemente su eficiencia con respecto a la versión 4.x la cual sólo tenía 80 MB.

Un servidor SQL mantiene las conexiones con de los usuarios, reservándoles memoria para la recuperación de datos. Obviamente si la página de datos está presente en memoria esto es mucho más rápido para traer los datos que si estuvieran en disco.

Sybase maneja un procesamiento en línea (OLTP), en el que la transferencia de datos se realiza básicamente por un número pequeño de renglones, sin esperar un tiempo significativo entre las operaciones de transacción, es decir la naturaleza de OLTP es grabar datos de manera rápida.

Sybase tiene su propio espacio el cual se puede ver como un pequeño mundo desconocido para Unix (o VMS).Aunque necesita comunicarse con el mundo externo, para ello utiliza dispositivos.

Todos los dispositivos son mapeados como dispositivos físicos esto


SQL:: Lenguaje de consulta y acceso a base de datos.


SQL-Server: Servidor de la Base de Datos desarrollado por Sybase.


Memoria Cache: Tiene como objetivo suministrarle los datos al procesador a la velocidad que los solicita.

es, un dispositivo físico (como puede ser /dev/rdsk/c0t1d0s4) es mapeado como un lógico (logdev). Dependiendo del tipo de dispositivo (disco, cinta, etc.).

El servidor SQL está compuesto por dispositivos de datos y dispositivos de transacción, estos dispositivos deben estar montados sobre particiones crudas, la razón de esto es porque Sybase necesita garantizar que todas las escrituras tengan un completo éxito. En el caso de que se monten sobre el sistema de archivos (File System) de Unix y la máquina se apague, Sybase no esta consciente de que los datos fueron escritos a disco. Esto fue porque la integridad de la base de datos se perdió. Por esto Sybase necesita de particiones crudas ya que solo así puede asegurar la integridad en los datos.

SQL y Transact-SQL

SQL es una herramienta para organizar gestionar y recuperar datos almacenados en una base de datos. El nombre SQL es una abreviatura de Structure Query Language (Lenguaje de Consulta Estructurado).

En junio de 1970 el Dr. Codd publicó un artículo titulado "Un Modelo Relacional de Datos para grandes bancos de datos compartidos" que esquematizaba una teoría matemática de como los datos podrían ser almacenados y manipulados utilizando una estructura tabular. Las bases de datos relaciónales y SQL tienen sus orígenes en este artículo.

SQL

Uno de los desarrollos más importantes en la aceptación del mercado de SQL es el surgimiento de los estándares SQL. El trabajo en el estándar SQL oficial comenzó en 1982 cuando la American National Standards Institute (ANSI) encargo a su comité X3H2 que definiera un lenguaje de

bases de datos relacional. Como SQL emergió como el estándar en el mercado el comité seleccionó SQL como su lenguaje de base de datos relacional y lo estandarizó.

El estándar ANSI para SQL resultante está basado en gran medida en el SQL del DB2, aunque contiene algunas diferencias importantes con respecto a él. El estándar fue oficialmente aceptado como estándar ANSI X3.135 en 1986 y como estándar ISO en 1987. El estándar ANSI/ISO ha sido adoptado desde entonces como estándar del Federal Information Processing Standard (FIPS) por el gobierno de los Estados Unidos.

Una aplicación que utiliza SQL puede funcionar indistintamente con cualquier sistema de gestión de bases de datos basado en SQL.

Front-End: En ambientes de bases de datos el software que le presenta la información al usuario.

Las funciones de un Data Base Manager System (DBMS) están divididas en dos partes. Los "frontales" (***front-ends***) de base de datos tales como herramientas de consulta interactiva escritores de informe y programas de aplicación, se ejecutan en la computadora personal. La máquina de soporte (***back-end***) de la base de datos que almacena y gestiona los datos se ejecuta en el servidor.

Back-End: Software o hardware que actúa sin ser visto en una arquitectura cliente/ servidor.

SQL se ha convertido en el lenguaje de base de datos estándar para comunicación entre las herramientas frontales y las máquinas de soporte.

SQL se utiliza para controlar todas las funciones que un DBMS proporciona a sus usuarios:

- ***Definición de datos***. SQL permite a un usuario definir la estructura y organización de los datos almacenados las relaciones entre ellos.
- ***Recuperación de datos***. SQL permite a un usuario o a un programa de aplicación recuperar los datos almacenados de la base de datos y utilizarlos.

Manipulación de datos. SQL permite a un usuario o a un programa de aplicación actualizar la base de datos añadiendo nuevos datos,

suprimiendo datos antiguos y modificando datos previamente almacenados.

- ***Control de acceso***. SQL puede ser utilizado para restringir la capacidad de un usuario para recuperar, añadir y modificar datos, protegiendo así los datos almacenados frente a accesos no autorizados.
- ***Compartición de datos***. SQL se utiliza para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieran unos con otros.
- ***Integridad de datos***. SQL define restricciones de la integridad en la base de datos, protegiéndola contra corrupciones debidas a actualizaciones inconsistentes o a fallas del sistema.
- En muchos sentidos, las consultas son el corazón del lenguaje SQL. Todas las sentencias SQL comienzan con un verbo (Ver Fig. 4.3), una palabra clave que describe lo que la sentencia hace CREATE, INSERT y DELETE son verbos típicos, la sentencia continúa con una o más cláusulas, una cláusula puede especificar los datos sobre los que debe actuar la sentencia, o proporcionar más detalles acerca de lo que la sentencia se supone que hace. Todas las cláusulas comienzan también con una palabra clave, tal como WHERE, FROM, INTO y HAVING. Algunas cláusulas son opcionales, otras, son necesarias. La estructura y contenido específicos varían de una cláusula a otra. Muchas cláusulas contienen nombres de tablas o columnas específicos, algunas pueden contener palabras claves adicionales, constantes o expresiones

Fig 4.3 Estructura de una sentencia SQL

A continuación se explicarán las sentencias básicas de SQL: SELECT, FROM, WHERE, GROUP BY, ORDER BY, HAVING.

SELECT : recupera datos de una base de datos y los devuelve en forma de resultados de la consulta. Se pueden especificar todos los campos de la tabla mediante el símbolo asterisco(*) o solo alguna parte poniendo el nombre de uno o varios campos que se deseen recuperar.

FROM : indica la tabla de la cual se extraerá la información

WHERE: Forma una condición para recuperar solo los datos que cumplan la condición establecida.

GROUP BY: agrupa por el nombre del campo que establezca formando grupos.

ORDER BY: ordena los datos de la tabla por el campo que se indique.

HAVING: utiliza una condición de búsqueda para especificar los grupos deseados.

ejemplo:

```
Select * from academico where rfc like = " [Aa] %" group by cvecat order by rfc
```

Recupera todos los campos de la tabla academico en donde el rfc comience con A o a agrupando por la clave de la categoría y ordenando por rfc.

Transact -SQL

Transact - SQL es una extensión de SQL propia de Sybase. Incluye las siguientes extensiones:

- variables
- condicional de ejecución(if..then)
- ciclos (while)
- manejo de errores

variables

La variables existen dentro de un procedimiento, son usadas en procedimientos almacenados y se declaran de la siguiente manera:

declare {@nombre_varable} {tipo_de _variable}

ejemplo

declare @nombre char(10)

Para asignar un valor a una variable se utiliza la sentencia select

ejemplo

select @total = count(*) from academico

Así como para desplegar el valor de una variable local se utilizan dos formas. La primera es usando la sentencia select y la segunda es utilizando la sentencia print.

Ejemplo

declare @titulo

select @titulo (primera forma)

print @titulo (segunda forma)

condicional de ejecución

La condicional realiza ciertas sentencias dependendiendo de la expresión booleana. Si es verdadera se realiza el primer conjunto de sentencias de lo contrario se lleva a cabo el segundo conjunto de sentencias.

La sintaxis se muestra a continuación:

```
If {expresión_booleana}
   {sentencias}
else
    {sentencias}
```

Las sentencias boolenas pueden utilizar los siguientes operadores de comparación:

```
>    (mayor que
!=   (diferente de)
```

Un conjunto de sentencias comienzan con **begin** y terminan con la palabra **end**

ejemplo

```
declare @acad int
   select  @acad = count(*) from academico
   If  @acad > 88000
      select " Existen más de 88000 nombramientos en la nómina "
   else
        select "No existen más de 88000 nombramientos en la nómina"
```

ciclos(while)

La estructura de control while realiza un conjunto de sentencia mientras la expresión booleana sea verdadera.

```
while {expresion_booleana}
   sentencias
```

ejemplo:

```
declare @cont int , @acad int
select @cont = 1°
```

```
while  @cont< = 3
        begin
                select @acad  =  count(*)  from proyecto
                where duracion =  @cont
                select @cont = @cont + 1
        end
```

En este ejemplo cuenta los investigadores que existen en la tabla proyecto con una duración de 1, 2 y 3 años.

manejo de errores

No existe un comando para abortar los errores. El formato de un error es el siguiente:

raiserror { numero de error} {carácter de error}

raiserror { numero de error} {variable local }

El número de error en Transact SQL puede ir del 1 hasta el 20000. El @@error.

Sybase define los códigos de error del 1 hasta 20000, el número de error y descripción son guardados en la base de datos llamada sysmessages en la base de datos master de error.

ejemplo

```
select error, description
from master..sysmessages
where description like '%duplicate%'
and ( description like '%key%' or description like '%index%' )
```

Muestra todos los mensajes de error que tengan la palabra duplicate o key o index en su descripción.

SQL es un lenguaje fácil de entender y una herramienta completa para gestionar datos. Para finalizar mencionaremos las principales características de SQL.

- Portabilidad a través de sistemas informáticos.
- Los estándares de SQL
- Su independencia de los vendedores
- Fundamento relacional
- Su estructura de alto nivel
- Su acceso a la base de datos mediante programas
- La arquitectura cliente/servidor
- Su definición dinámica de datos
- las vistas múltiple de datos

Respaldos

Es muy importante que periódicamente se respalde la información, esto es, que se tenga duplicado de la misma. En esta forma, si por alguna causa se llegase a dañar el lugar donde se aloja la información (por ejemplo, el disco duro), se tendría como opción recuperar la información de algún otro lugar (por ejemplo una cinta).

Nosotros usaremos el nombre lógico **backcinta** como dispositivo de respaldo en cinta, de tal manera que si quisiéramos respaldar nuestra base de datos *papiit* en una cinta, tendríamos que teclear lo siguiente dentro de una sesión de SQL:

```
1 > dump database papiit to backcinta
2 > go
```

El administrador Sybase es la persona encargada de realizar estos respaldos.

Concluyendo con este capítulo nos remitiremos a explicar brevemente la creación de la base de datos PAPIIT la cual se reliazó a través de SCRIPTS utilizando el lenguaje de SQL-Server (Transac SQL) para la creación de las Tablas que conforman la base de datos así como las vistas a tablas a esta.

Realizando asimismo SCRIPTS para el borrado de tablas e inserción de datos tanto de prueba como reales a estas tablas.

A continuación se muestra parte de los scripts realizados:


Scrip: Conjunto de instrucciones en SQL.

Script para la creación de tablas y vistas

```
use hpapiit        " pone en uso la base de datos en la sobre la que se crearan las
                     vistas y tablas
go                 " ejecuta la instrucción
```

Es importante hacer notar que esta vista fue creada a partir de la base de datos de la Nómina General de la UNAM de nominada nomina

```
print "VISTA academico " " envia un mensaje a pantalla mientras se ejecuta la instrucción
create view academico as
select rfc,nombre,cvecat,cvedep,nomcat,horas from nomina.sistemas.academico
go
print"TABLA COMITE "
create table comite(cvecom char(2) NULL,nomcom char(50) NULL,maximo tinyint NULL)
go
print "TABLA PROTOCOLO"
create table protocolo5 (cveproy char(8) NULL,antecedentes text NULL,
                objetivos text NULL,hipotesis text NULL,
                metas text NULL,metodologia text NULL,
                infraestructura text NULL)
go
print"TABLA PROYECTO "
create table proyecto5 (cveproy char(8) NULL,nomproy char(250) NULL,
                        cvedis char(4) NULL,cvecom char(2) NULL,
                       cvedep char(5) NULL,
                        duracion tinyint NULL,
                        inicio tinyint NULL)
go
....
```

Script para la inserción de datos en tablas

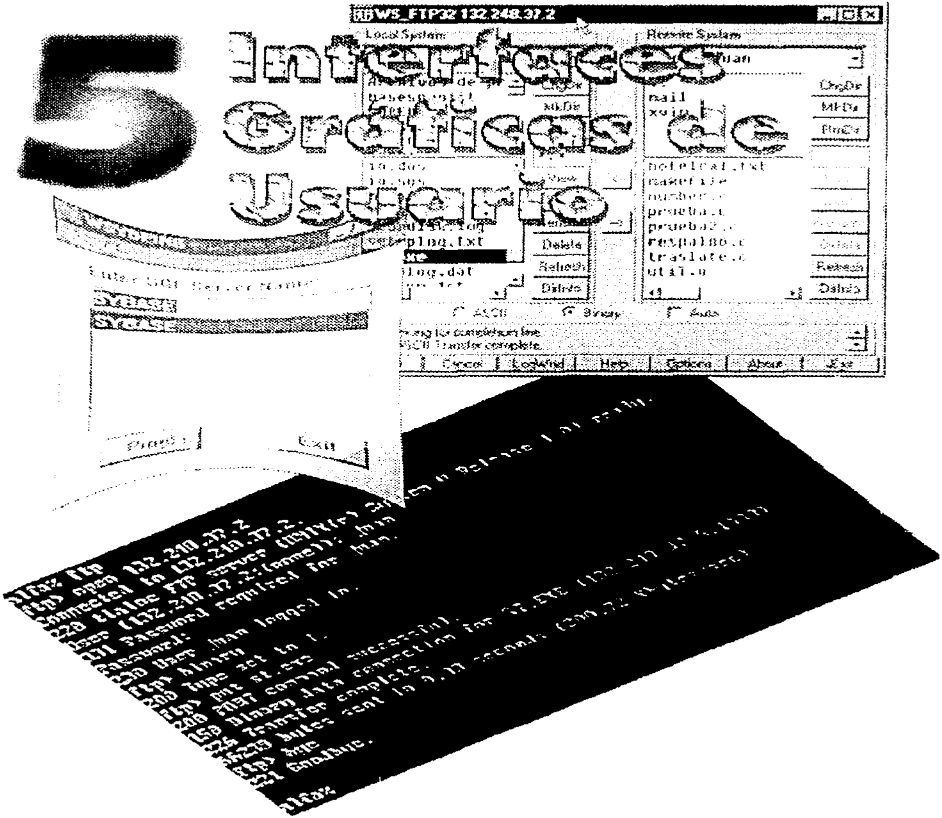
```
print "USANDO PAPIIT"
use papiit
go
print "INSERTA EN LA TABLA COMITE"
insert into comite values("1","Ciencias Físico-Matemáticas y de las Ingenierías","2")
insert into comite values("2","Ciencias Biológicas y de la salud","3")
```

```
insert into comite values("3","Ciencias Sociales","2")
insert into comite values("4","Humanidades y Artes","3")
insert into comite values("5","Innovación Tecnológica","2")

print "INSERTA EN LA TABLA PROTOCOLO"
insert into protocolo values ("IN111194","La estructura molecular de las encimas del frijol presentan rugosidad en la primera etapa de crecimiento", "Proporcionar datos fidedignos que comprueben que esto se debe a que la semilla del frijol no se encuentra en las condiciones adecuadas para un crecimiento adecuado", "La rugosidad presentada se debe a que la tierra contiene una cantidad menor de proteínas","Deteminar la cantidad adecuada de proteínas que debe contener la tierra para evitar esta rugosidad","1.Observar al frijol desdes que es plantado 2.Plantar frjol con las mismas características en diferentes tierras cada una con diferentes niveles de proteínas 3. Medir el cremiento por dia de la planta del frijol, 3.Determinar en que día comienzan aparecer la rugosidad en el frijol","Se cuebta con un invernadero con las condiciones adecuadas, así como con todo el material necesario para el desarrollo del proyecto,")
go
print "INSERTA EN LA TABLA PROYECTO"
cveproy,nomproy,cvedis,cvecom,cvedep,duracion,inicio
insert into proyecto values("IN111194","Rugosidad en el frijol","0200","2","47301","3","94")
insert into proyecto values("IN200697","Las amibas en vegetales"0100","1","48101","3","97"
insert into proyecto values("IN300297","Los virus informáticos","01004","3","42901","3","97"
insert into proyecto values("IN200897","Cantidad proteinica en el trigo","0200","5","33901","3","97"
go
....
```

Script para el borrado de tablas

```
print "USANDO PAPIIT. .."
use papiit
print "TABLA COMITÉ"
drop table comité
go
print "TABLA PROTOCOLO"
drop table protocolo
go
print"TABLA PROYECTO"
drop table proyecto
go
...
```

5 Interfaces Gráficas de Usuario



A través del tiempo el hombre siempre ha tenido la necesidad de comunicarse entre sí, esto de la manera más simple y clara dado la importancia que implica expresar lo que pensamos y sentimos, esto en la rama de la computación es muy importante y ha dado pie a la creación de sistemas amigables basados en gráficas que le permiten a los usuarios realizar sus tareas sin muchas complicaciones. De esta manera, el uso de un equipo de cómputo no queda restringido sólo a un grupo de usuarios expertos.

El origen de las Interfaces Gráficas de Usuario (GUI: Graphics User Interface) radica en la necesidad de tener un ambiente de trabajo más sencillo para el usuario y evitar el manejo complicado de comandos así como su difícil memorización.

Interface Gráfica de Usuario (GUI: Graphic User Interface)

La GUI, como su nombre lo indica, emplea gráficos para comunicarse con el usuario. Las GUI suelen llamarse interfaces WIMP. Este acrónimo se refiere a "Windows Icons Menus Pointing device" ó "Ventanas Iconos Menús Dispositivos de señalamiento (puntero ó mouse)" y, en general, todos estos elementos se encontrarían en una GUI. La figura 5.1 muestra los componentes de una GUI.

Las GUI permiten que el usuario divida la pantalla de una computadora en ***ventanas*** sobrepuestas o mosaicos. Puesto que una aplicación no ocupa la pantalla completa, es posible mostrar varias aplicaciones al mismo tiempo, si bien el tamaño de cada ventana puede variar y una ventana puede cubrir temporalmente otras ventanas.

Una ventana es una área de la pantalla de la computadora que contiene la salida de una aplicación.

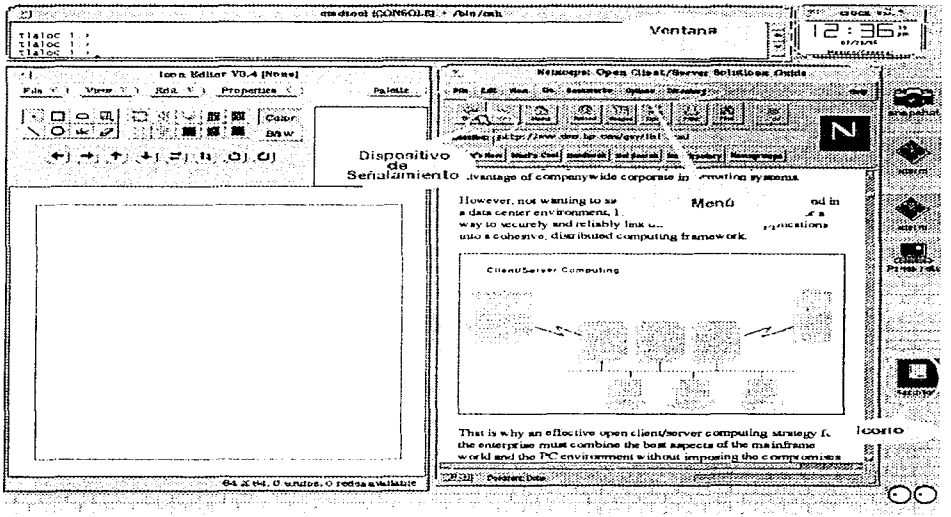


Fig 5.1. Componentes de una Graphic User Interface (GUI)

Hablar de interfaces gráficas de usuario, ayuda a pensar en términos de lo que se conoce como "la metáfora del escritorio": la pantalla es el equivalente a este mueble de oficina. Cuando se concertan citas, la agenda es la que está arriba en el escritorio, pero si se necesita buscar un número telefónico, el directorio se pone encima de los demás documentos, cubriendo todo lo que está abajo. Una vez que se encuentra el número deseado, el directorio puede apartarse, haciendo que lo que está abajo del mismo sea el documento que quedará arriba de todos los demás. Si lo desea, puede irse más abajo de la agenda, tomar una hoja de papel cubierta de documentos y ponerla encima de todos estos, haciendo foco de la atención. Una GUI funciona de la misma manera.

La capacidad para dividir la pantalla en ventanas aumenta la productividad del usuario. Así, un usuario puede anotar en una ventana las consultas realizadas a una base de datos y visualizar los resultados de consultas anteriores en otra, evitando la necesidad de sentarse a esperar que las consultas se procesen.


Los iconos son pequeñas imágenes.

En una GUI la forma de introducir comandos es mucho más sencillo que en sistemas basados en caracteres. El usuario hace sus elecciones señalando *iconos* ó eligiendo la acción deseada de una lista, en lugar de teclear órdenes en la línea de comandos del sistema. De ser necesario, es posible utilizar menús a fin de visualizar las opciones disponibles; al mismo tiempo, aparecen ventanas en respuesta a las opciones más complejas del menú, en las cuales se solicita al usuario que anote datos adicionales o que seleccione opciones adicionales. Este enfoque basado en gráficos proporciona al usuario bastante más información que una interface basada en caracteres, haciendo que la interface sea más intuitiva y, por consiguiente, más sencilla de aprender.

Razones para usar una GUI

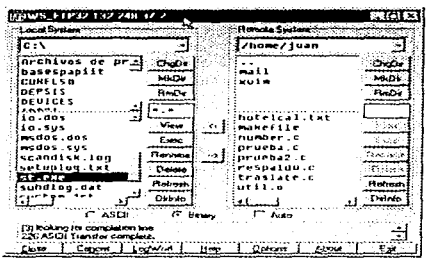
Quien haya usado alguna vez una computadora personal habrá utilizado una interface basada en caracteres. De hecho, casi todas las aplicaciones de software para computadoras personales de mayor popularidad tienen una interface basada en caracteres. Durante algún tiempo los fabricantes de software se resistían a la idea de hacer un cambio a las GUI's por las siguientes razones: a) La aplicación debería escribirse casi por completo para que pudiera correrse en un ambiente gráfico y b) Las GUI's le exigen al programador que siga ciertas reglas en la interface del usuario, haciendo que las aplicaciones GUI sean diferentes a sus hermanas basadas en caracteres. Sin embargo, las ventajas de las GUI son ahora tan evidentes que los fabricantes de software de mayor popularidad tienen versiones GUI de sus aplicaciones.

Facilidad de uso

Las persona pueden procesar más información visualmente que por medio de cualquier otro sentido y una GUI saca mucho provecho de esto. Así, un usuario encuentra que una GUI es a la vez más sencilla de aprender y más sencilla de utilizar que una interface basada en caracteres.

Las personas que hayan usado un sistema operativo basado en caracteres (por ejemplo MS-DOS) habrán tenido bastantes experiencias en este sentido, cuantas veces no habrán tecleado "dor" o "der" en lugar del comando "dir"; por el contrario, las interfaces gráficas no presentan estos problemas, ya que las operaciones que puede realizar un usuario vienen en representadas en la pantalla de la computadora con imágenes o "botones".

(a)



(b)

Fig. El uso de aplicaciones gráficas es más sencillo e intuitivo; (a) El programa FTP en modo carácter requiere del conocimiento de varios comandos para ser usado, (b) El programa FTP para windows muestra en pantalla las opciones del usuario, sin que tenga que memorizarlas.

Consistencia con los diferentes sistemas operativos

Otra ventaja de las GUI's es su consistencia con los diferentes sistemas operativos. Puesto que las GUI's siguen un estándar, las aplicaciones GUI's tendrán el mismo funcionamiento en cualquier sistema operativo, con lo que el usuario no tendrá problemas en usar cualquier sistema operativo. La figura 5.2. muestra dos GUI en diferentes sistemas operativos.

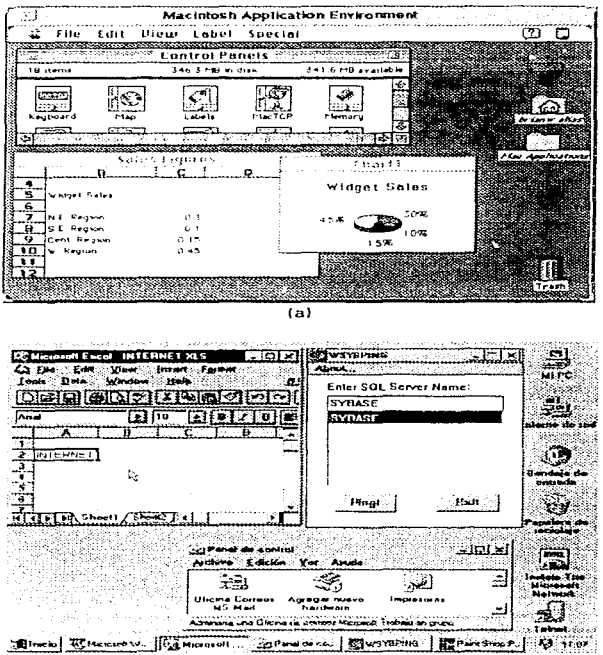
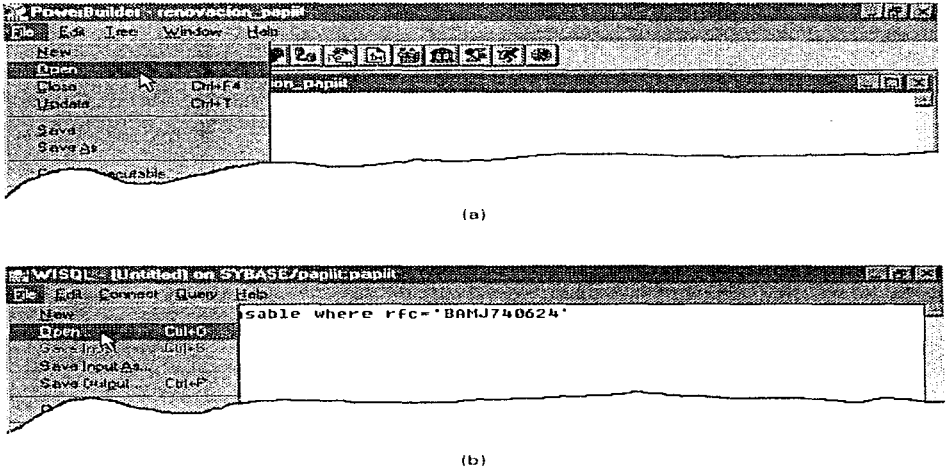


Fig. 5.2. Las GUI's en diferentes sistemas operativos, (a) sistema operativo Macintosh y (b) sistema operativo Windows 95. Se puede observar que el manejo de ambas GUI es intuitivo y muy parecido.

Consistencia entre aplicaciones

Una ventaja más ofrecida por las GUI es la consistencia entre las aplicaciones. Esta consistencia le permite al usuario ser más productivo en menos tiempo. Esto es que, los usuarios sabrán, sin remitirse al manual, que siempre deberán llamar el menú "File" para abrir un archivo y el menú "Edit" para copiar datos de una aplicación y pasarlos a otra. Esta consistencia en la estructura de los menús hace mucho más sencillo que el usuario final aprenda nuevas aplicaciones, ya que las funciones que son comunes en aplicaciones tienen el mismo nombre y se manejan de manera consistente. Ver fig. 5.3.



(a)

(b)

Fig. 5.3. La consistencia entre las GUI's hace que los usuarios tengan un aprendizaje más rápido, (a) menú de la aplicación PowerBuilder, (b) menú de la aplicación WISQL.

Intercambio de datos de alto nivel entre aplicaciones

Un clip electrónico o portapapeles es una parte de la memoria de la computadora donde se almacenan texto, figuras o gráficas

Aun cuando en sentido estricto no forman parte de las GUI's, la mayoría de estas interfaces proporcionan a los usuarios formas simples para transferir información de una aplicación a otra. Para elic se utiliza un ***clip electrónico*** o mejor conocido ***portapapeles***. Si el usuario desea transferir datos (texto, figuras ó gráficas) de una aplicación a otra, lo único que tiene que hacer es "copiarlos" al clip electrónico cuando está corriendo la primera aplicación y después de correr la segunda aplicación, puede "pegar" los datos del clip electrónico al lugar que el desee. Con esta opción el usuario podrá ahorrar mucho al "copiar" un trabajo que ya había hecho, en lugar de hacerlo nuevamente (un caso común es el procesamiento de texto, en vez de capturar todo un texto que ya se había capturado previamente o existe en algún otro documento, sólo se "copia"). Ver fig. 5.4.

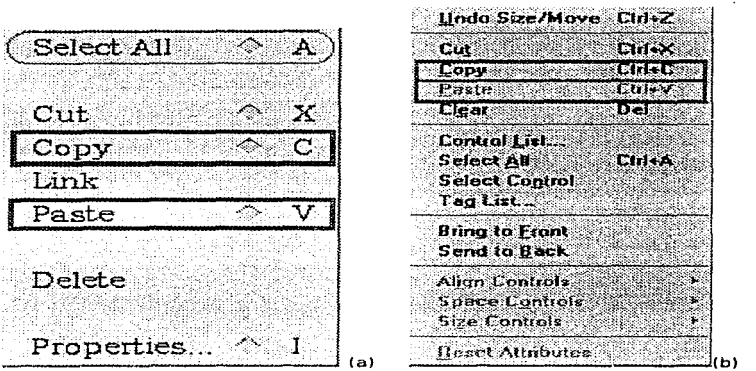


Fig. 5.4. La transferencia de datos entre aplicaciones se puede realizar a través de un clip electrónico usando la opción de "copiar" y "pegar", (a) opciones de un clip electrónico de una aplicación en una estación de trabajo, (b) opciones de un clip electrónico de una aplicación en una computadora personal.

Programación con GUI's

Podemos ver a las GUI's desde dos puntos de vista:

- Desde el punto de vista del usuario y
- Desde el punto de vista del programador

Desde el *punto de vista del usuario*, el sólo ve ventanas, iconos, menús e interactua con ellos mediante el uso de un dispositivo de señalamiento (ratón); el usuario utiliza la GUI para realizar una actividad específica, como por ejemplo, consultar la información relacionada con las solicitudes de inscripción al Programa de Apoyo a Proyectos de Innovación e Investigación Tecnológica (PAPIIT) y no tiene que conocer nada relacionado con la programación que esta detrás de la aplicación; el usuario sólo debe necesita conocer el funcionamiento de la aplicación GUI que está utilizando, tarea que no es difícil dadas las bondades que ofrecen de las GUI's. Ver figura 5.5.

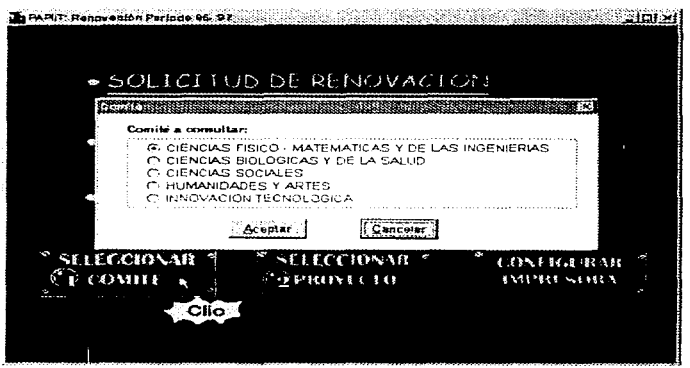


Fig. 5.5. En una aplicación GUI el usuario no tiene que saber conceptos de programación.

Desde el punto de vista del programador, por el contrario, se debe de tener conocimiento del lenguaje de programación que se usa para el desarrollo

de la aplicación GUI y conocer los elementos necesarios para programar. El programador es el encargado de escribir el código fuente que se necesita para que la aplicación GUI funcione, y funcione adecuadamente. El programador es responsable de establecer las opciones a las que tiene acceso un usuario y prever las posibles combinaciones de acciones que pueda realizar un usuario, de tal manera, que la aplicación no resulte en una falla. Ver figura 5.6.

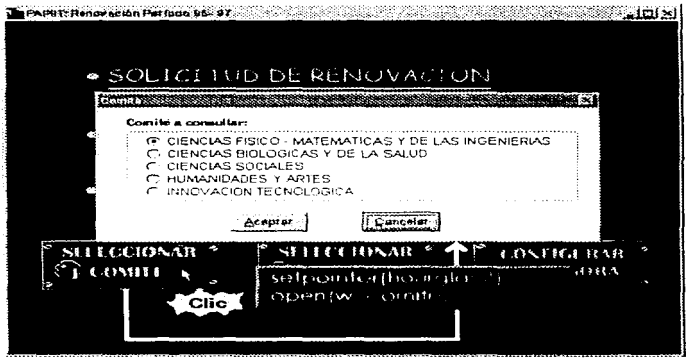


Fig. 5.6. El programador necesita conocer el lenguaje de programación de la aplicación y establecer la forma en que funcionará la GUI.

Programación Orientada a Objetos

Como antecedente veremos un panorama general de la Programación Orientada a Objetos (POO).

La Programación Orientada a Objetos es el análisis, diseño e implementación de sistemas vistos desde la perspectiva de *objetos.*, los cuales tienen *atributos* específicos y *eventos* asociados con estos.

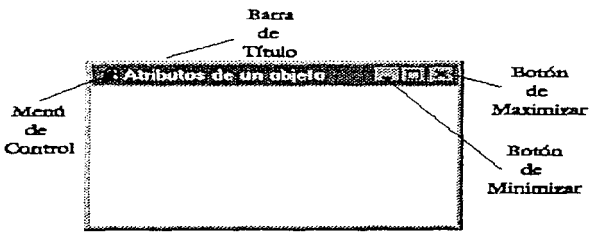
Los *objetos* son componentes de la aplicación que combinan características y conductas. Las características son llamadas *atributos*, las

conductas son llamados *eventos*. Algunos ejemplos de objetos en PowerBuilder son: window (ventana), menu (menú), project (proyecto), etc.

Los objetos se comunican unos a otros por medio de *mensajes*. Un objeto puede enviar un mensaje que contenga algunos datos a otro objeto, sin saber que es lo que va a hacer el objeto que reciba el *mensaje* con los datos. El objeto acepta que va a recibir el *mensaje* acepta y sabe que es lo que va a hacer. Ver fig. 5.6 donde una ventana se comunica con otra a través de un *mensaje*.

Los *atributos* son las características de los objetos. Principalmente, los atributos de un objeto determinan como se ve éste. Algunos *atributos*, sin embargo, establecen la conducta para un objeto.

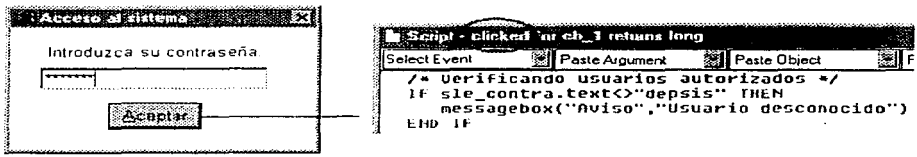
Por ejemplo, algunos de los *atributos* de una ventana son el color, el tamaño, el texto del título, la presencia o ausencia de los botones de maximizar y minimizar, y cuando ésta tiene un menú de control. Estos atributos ayudan a definir la apariencia de la ventana.



En este caso los botones de maximizar y minimizar, el menú de control especifican cierta conducta para la ventana; es decir, ellos le permiten a ésta estar maximizada, minimizada y cerrada, respectivamente.

Un *evento* es el componente de un objetos que le permite a éste reconocer y responder a mensajes. En PowerBuilder tu escribes *scripts*, cuando es necesario, para describir el procesamiento que debería ocurrir en respuesta a un mensaje.

Por ejemplo, cuando se quiere verificar el accesc a la aplicación, el evento "Clicked" de un *botón de comando* podría mandar un mensaje de usuario desconocido en caso de que la contraseña de acceso sea incorrecta.



Los scripts se pueden escribir dando clic derecho sobre el objeto y escoger la opción *script..* del menú mostrado, o bien dando un clic sobre el painter de script ( ).

Con PowerBuilder, así como con otras herramientas como Visual Basic, Delphi, Neuron Data, etc., se pueden crear aplicaciones que aprovechan la Interface Gráfica de Usuario (GUI) de una manera sencilla.

Porqué usar PowerBuilder

Se usará PowerBuilder por la siguiente razón: PowerBuilder resultó ser el mejor software de una serie de pruebas aplicadas a una lista de software's similares, las pruebas incluían:

- Conectividad a fuentes de datos
- Manipulación de datos
- Facilidad de programación
- Conversión de datos

Programación con PowerBuilder

PowerBuilder es una herramienta orientada a objetos usada para el desarrollo de aplicaciones Cliente/Servidor. Esta permite a uno o más desarrolladores construir aplicaciones gráficas fácil y rápidamente que accesan información de bases de datos alojadas localmente o en servidores en red.

A los programas como PowerBuilder se les conoce como *clientes* o *front-ends*. Los *front-ends* recuperan información y la presentan al usuario de muchas maneras: en forma de reportes, gráficas, columnas, etc.

PowerBuilder utiliza para el desarrollo de aplicaciones básicamente un lenguaje de programación basado en lenguaje C y C++ conocido como *PowerScript*. Al conjunto de instrucciones dentro de un evento se le llama *script*.

Variables de ambiente

En nuestra aplicación, PowerBuilder utilizará la base de datos Sybase; por tanto, es indispensable tener instalado Open Client for Windows y tener definidas las siguientes variables de ambiente en nuestro en nuestra computadora:

```
LANG = enu
SYBASE = C:\SQL10
DSQUERY = SYBASE
PATH = C:\SQL10\BIN;C:\SQL10\DLL
LIB = C:\SQL10\LIB
INCLUDE = C:\SQL10\INCLUDE
```

Si no se tienen definidas, se puede crear el siguiente archivo de texto **wsybset.bat**, en el directorio c:\sql10\bin, que contenga las siguientes líneas:

```
REM  Variables de ambiente para Open Client for Windows
SET LANG = enu
SET SYBASE = C:\SQL10
SET DSQUERY = SYBASE
SET PATH = C:\SQL10\BIN;C:\SQL10\DLL;%PATH%
SET LIB = C:\SQL10\LIB;%LIB%
SET INCLUDE = C:\SQL10\INCLUDE;%INCLUDE%
```

e incluir la siguiente línea en nuestro archivo de arranque *autoexec.bat*:

```
PROMPT $p$g
...
CALL  c:\sql10\bin\wsybset.bat
...
```

Una vez guardados los cambios en el archivo *autoexec.bat* es necesario reiniciar la computadora para que los cambios tengan efecto.

Entorno de programación PowerBuilder

Describiremos brevemente el entorno de programación de PowerBuilder:

Barra de título: Muestra el título referente al contenido de la ventana.

Barra de menú: Muestra las opciones que el usuario puede realizar en la ventana activa (la aplicación puede tener más de una ventana, pero sólo una está activa).

Barra de painters: También conocida como barra de herramientas, está conformada de varios "botones" que realizan una acción rápidamente, sin necesidad de utilizar el menú (todas las acciones realizadas por los "botones" incluidos en la barra painters se pueden realizar desde el menú).

Area del cliente: Es el área de trabajo del usuario (donde estarán abiertas las ventanas de trabajo).

Barra de estado: Es una barra que informa al usuario el estado de la aplicación (Imprimiendo..., Exportado datos..., etc. son ejemplos).

Los painters o "botones" representan a los objetos que se pueden crear con PowerBuilder, los más comúnmente usados son los siguientes:

 Painter Application

Especifica información acerca de tu aplicación, como es el nombre y librerías en las cuales los objetos de la aplicación van a residir.

 Painter Project

Crea un archivo ejecutable de tu aplicación.

 Painter Window

Construye las ventanas que tu vas a usar en la aplicación.

 Painter User Object

Construye objetos personalizados que tu puedes salvar y reusar repetidas veces en ventanas.

 Painter Menu

Construye los menús que las ventanas van a utilizar.

 Painter Datawindow

Construye la interfaz con la base de datos.

 Painter ODBC (Object Data Base Connection)

Define una fuente de datos para archivos de texto (.txt), de dBase (.dbf), de Access (.md), etc.

 Painter Database

Especifica la forma de conectarse a la base de datos.

 Painter Pipeline

Copia datos de una base de datos a otra (no importando el tipo de bases de datos).

 Painter Library

Mantiene *librerías* de PowerBuilder; permite copiar, mover y eliminar objetos. Una *librería* es un archivo con extensión .pbl que almacena los objetos (como ventanas, menús, application, datawindow, etc.). Las *librerías* pueden compartirse entre programas, esto es, un programa puede ocupar más de una librería.

 Painter Edit

Editor de archivos.

 Painter Run

Ejecuta la aplicación actual (muestra el funcionamiento de la aplicación).

 Painter Debug

Corre paso a paso una aplicación (nos sirve para detectar errores de programación).


Painter Exit
Finalización de PowerBuilder.

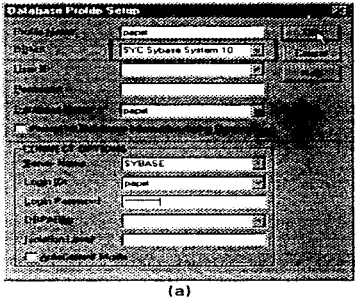
A continuación haremos mostraremos las características más relevantes de la programación con PowerBuilder:


Los ejemplos mostrados en este capítulo están escritos en el lenguaje PowerScript de ***PowerBuilder*** en su versión 5.0.

- **Un ambiente para aplicaciones gráficas independiente del dispositivo**

Por lo general, las GUI's ofrecen características de compatibilidad con una amplia variedad de dispositivos de hardware y con una amplía variedad de sistemas manejadores de bases de datos, eliminando la necesidad de que los programadores de software se ocupen de estos aspectos. Por ejemplo, una GUI puede interactuar con distintas fuentes de datos simultáneamente, como pueden ser un servidor de bases de datos Sybase System 10, Oracle, Informix, GUPTA, etc., o archivos de texto, o un archivos dbf; sin que el programador tenga que conocer el funcionamiento propio de la fuente de datos. Ver fig. 5.7.


Al presionar este painter, y después la opción del menú
File -> Connect... -> Setup...
se puede establecer una conexión a una base de datos.



(a)

```
/*
** Conexión a una fuente de datos
** Este script debe de ir en el evento "Open" del objeto
** Application (SQLCA)
*/
// Atributos del objeto Application
SQLCA.DBMS = "SYC Sybase System 10"
SQLCA.Database = "papiit"
SQLCA.LogID = "papiit"
SQLCA.Logpass = "papiit@admin"
SQLCA.ServerName = "SYBASE"
SQLCA.Autocommit = false

// Conexión a la base de datos papiit
Connect;
// Verificando conexión
If SQLCA.sqlcode < > 0 Then
        MessageBox("Aviso", "Fallo en la conexión")
        Halt Close
End If
// Desconexión a la base de datos papiit
Disconnect;
```

(b)

Fig. 5.7 En la realización de una aplicación el programador sólo tiene que seleccionar la fuente de datos que va a usar, sin conocer como trabaja ésta. (a) conexión una fuente de datos en desarrollo de la aplicación, (b) conexión a una fuente de datos cuando se está ejecutando la aplicación.

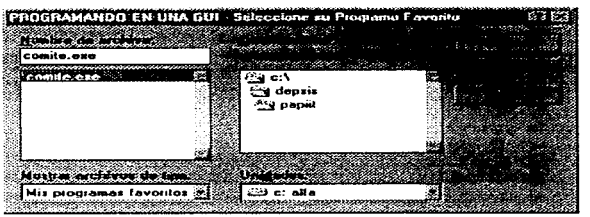
- **Una biblioteca de gráficos.**

Los programadores de software podrán recurrir a esta biblioteca de rutinas de acuerdo con sus necesidades, facilitándose así la elaboración de aplicaciones basadas en gráficos. Por ejemplo, no tendrán que escribir código

fuente para crear cajas de diálogo (abrir o guardar un archivo, mostrar un mensaje de error o ayuda, etc.), trazar formas específicas en la pantalla (círculos, rectángulo, líneas etc.) o seleccionar colores, puesto que las rutinas de software necesarias se encuentran ya construidas en la biblioteca y sólo tienen que ser utilizadas por medio de algunas pocas líneas de código. Ver. fig. 5.8.

```
/*
** Código fuente para abrir un programa existente (la calculadora de windows,
** por ejemplo)
*/
/* El símbolo & indica que la instrucción continua en la siguiente línea */
string ruta, doc
GetFileOpenName("PROGRAMANDO EN UNA GUI - Seleccione &
                 su Programa Favorito", ruta, doc, "*.exe",&
                 "Mis programas favoritos" + "," + "*.exe")
```

(a)



(b)

Fig. 5.8. La disponibilidad de una biblioteca de gráficos facilita la tarea al programador. (a) código fuente para crear una ventana de selección de archivos, (b) resultado del script presentado.

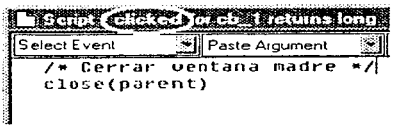
- **Medios interconstruídos para crear objetos**

Los programadores de software podrán desarrollar fácilmente aplicaciones que operen con objetos, pues una GUI cuenta con los medios para crearlos.

Los objetos window, menu, datawindow (que veremos más adelante) pueden tener una presentación preliminar, sin necesidad de ejecutar el programa, oprimiendo el botón de "preview" ( ) que aparecerá cuando se creen los objetos antes mencionados.

Las propiedades (color, tipo de letra, icono, cursor, etc.) de los objetos (application, window, menu, datawindow, etc.) se especifican dando un clic sobre el painter de propiedades ( ), o bien, dando un clic derecho sobre el y depués elegir *Properties...* ( ).

PowerBuilder, como dijimos anteriormente, usa el lenguaje de programación *PowerScript*, por lo que al escribir un script ( ) se **debe compilar**, para ver si no tiene errores de programación. La compilación con el painter de "Return" (es el último painter de la barra ubicada a la derecha), la apariencia del painter "Return" depende del script que se este escribiendo, si es el escript de una aplicacion este se vería así ( ), si es el de un objeto de una ventana se vería así ( ), etc. Un script tendría esta apariencia:



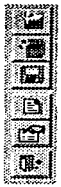
Un script le dice al objeto que hacer y en que momento (evento); por ejemplo, este es el script de un botón de Aceptar: en cuanto el usuario de un clic sobre éste, el cerrará la ventana madre.

Los objetos guardan desde la opción Save del menú File (File -> Save).

Una descripción de los painter de PowerBuilder más comunmente usados es la siguiente:

Crea objetos Application (aplicación)

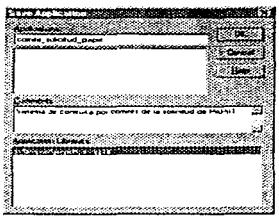
Por default PowerBuilder siempre tiene una aplicación abierta con las siguientes opciones:



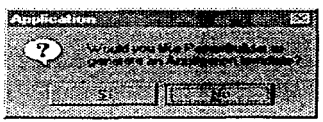
Nuevo
Abrir
Guardar
Script
Propiedades del objeto
Compilar el script (siempre es el último botón)

El objeto *application* es la **base de la aplicación** y es un objeto **no visible** (como una ventana, por ejemplo).

Para crear una nueva aplicación oprimimos el painter de "Nuevo", seleccionamos el directorio y nombre de la aplicación, y a continuación no preguntará una descripción de la aplicación:



Y enseguida, nos hará la siguiente pregunta:



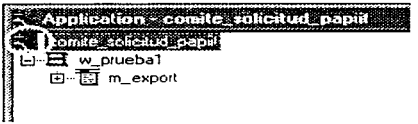
Deseas que PowerBuilder genere una aplicación patron?
Si No

Una aplicación patron es una aplicación con ventanas, menús y scripts predeterminados.

Podemos presionar "Si" para ver el efecto de esta acción, sin embargo, para nuestros ejemplos no es necesario, presionaremos "No".

Un ejemplo de un **script** típico en el objeto de application es el mostrado en la fig. 5.6 (b).

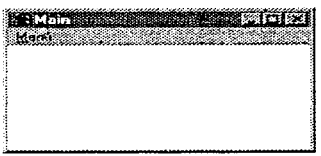
Un objeto application o aplicación es la base de un programa, a partir de este se derivan todos los demás objetos, como se puede ver a continuación:



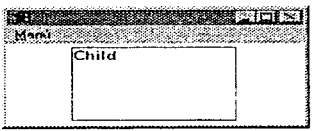
Para expandir los objetos dependientes de una aplicación se da doble clic sobre el icono de la aplicación (encerrado en un circulo). Y para abrir uno de los objetos se presiona <Enter> sobre el objeto.

Crea objetos window (ventana)

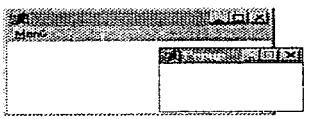
Las ventanas pueden ser de algunos de los siguientes tipos:



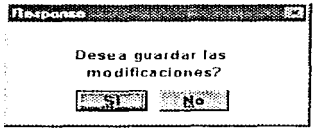
Main
Usada para iniciar aplicaciones SDI (Single Document Interface), es decir, de un solo documento. Un ejemplo de estas aplicaciones es la calculadora de windows.
Estás ventanas operan independientemente de otras ventanas.



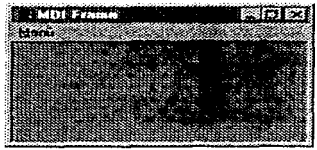
Child
Es utilizada para desplegar información.
Tiene una ventana "madre" y siempre estará dentro de los limites de ésta.
Cuando es minimizada se muestra el icono dentro de la ventana "madre".



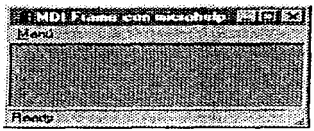
Popup
Es usada para ayuda en línea.
Tiene una ventana madre.
Puede ser desplegada fuera de la ventana madre.
Cuando es minimizada se muestra el icono fuera de la ventana "madre".



Response
Es utilizada para desplegar avisos o preguntas. Mantiene el control de la aplicación hasta que se de un clic a uno de los botones mostrados.

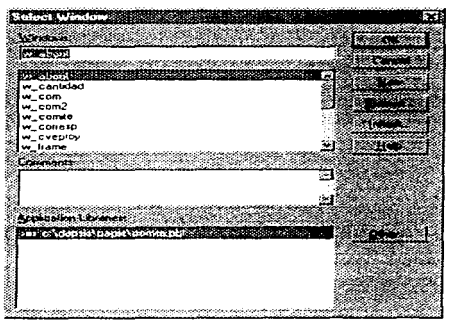


MDI Frame
Permite trabajar con múltiples documentos a la vez (MDI, Múltiple Document Interface) a través de una sola interface.



MDI Frame with microhelp
Al igual que la ventana MDI Frame permite trabajar con múltiples documentos a la vez, y además tiene una barra de estado para desplegar ayuda. PowerBuilder es una ejemplo de este tipo de ventana.

Al dar un clic sobre el painter ()nos mostrará la siguiente caja de diálogo:

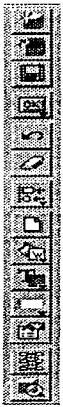


Abrir
Cancelar
Nueva
Buscar
Heredar (*ventana hija*)

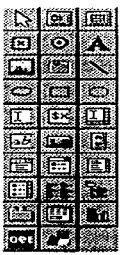
Buscar en otra Librería

Fig. 5.9. Selección de objetos

El resultado de elegir abrir, nueva o heredar (Inherit) son los siguientes botones:



Abrir
Nueva
Salvar
Objetos disponibles
Deshacer modificaciones
Eliminar objeto
Alinear objetos
Script
Color de relleno de los objetos
Color de letra
Borde del objeto
Propiedades del objeto seleccionado
Secuencia entre los objetos
Presentación preliminar de la ventana



Estos son los objetos disponibles para una ventana

Un ejemplo de una ventana creada algunos de estos objetos es la siguiente:

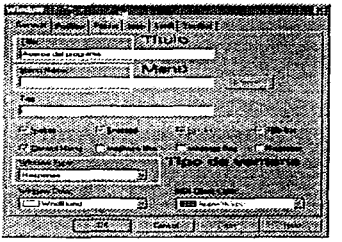
Todos los objetos "Aceptar" son susceptibles de tener **scripts**; por ejemplo, en esta ventana el **script** del botón "Aceptar" sería el siguiente:

```
// Este es un comentario
/*
Este es otro comentario
Más comentarios
*/
Close(Parent)
```

el cual indica que cierre la ventana donde está el mismo. Sin embrago, todos los objetos (inclusive la imagen puede tener un **script**).

Los comentarios en un script inician con // por cada línea, o bien, si se trata de un bloque de líneas inician con /* y finalizan con */.

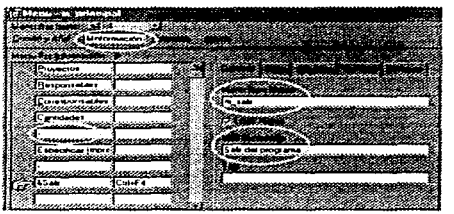
Por default, el tipo de ventana que da PowerBuilder es la de tipo main, si nosotros queremos cambiar el tipo de ventana tenemos que dar un clic al botón de propiedades ( ) de la ventana, o bien, dar doble clic sobre la ventana. Por ejemplo, las propiedades de la ventana mostrada arriba son las siguientes:



Esta ventana no tiene menú por ser una ventana de tipo **Response** (sólo muestra avisos o realiza preguntas).

Crea objetos menu (menú)

Al dar un clic sobre éste painter aparecerá una caja de diálogo semejante a la mostrada en la fig. 5.9. Si nosotros elegimos la opción "New" nos creará un menú en blanco. La forma de llenar las opciones de un menú es la siguiente:



El símbolo & sirve para subrayar una letra (&Información = Información), esto se usa para acceder la opción desde el teclado (Alt I).

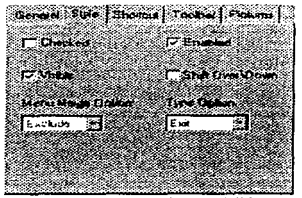
El símbolo - sirve para crear una división en el menú.

La "manita" indica la opción que esta activa.

El folder "General" indica el nombre de la opción del menú; los nombres siempre son en letras minúsculas e ignoran los acentos (&Información = m_informacin). Los nombre nos sirven para después acceder a las opciones del menú desde un script, para hacer esto es necesario indicar todos las opciones de menú anteriores; por ejmplo, si quisieramos deshabilitar la opción salir desde un script, este sería el siguiente:

```
m_principal.m_informacin.m_salir.enabled = false
```

También en este folder se escribe la ayuda en línea (MDI microhelp) que va a tener la opción.

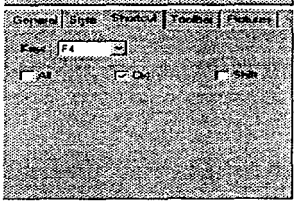


En el folder **Style** se indican los atributos de la opción del menú:

Checked - que tenga la marca ☑ al inicio del nombre de la opción.

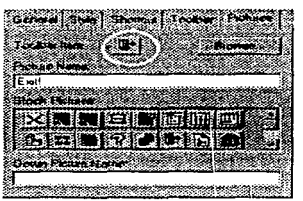
Enabled - que este habilitado

Visible - que se vea

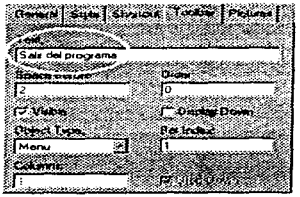


En el folder Shortcut se indica una combinación de teclas rápidas de acceso a la opción:

Salir = Ctrl + F4



En el folder **Pictures** se especifican los painters que aparecerán en la barra de herramientas de nuestra aplicación. Así se vería nuestra barra de painters:



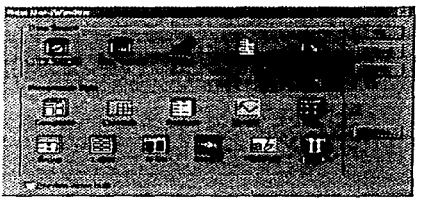
En el folder de **Toolbar** se especifica el texto que se vería cada vez que pase el cursor sobre el pinter. Así se vería:

Crea objetos datawindow

Un **datawindow** es un objeto que es usado para desplegar datos y/o capturar entradas del usuario.

Las *fuentes de datos* con las que un datawindow puede interactuar son muchas, por ejemplo; Sybase, Oracle, Informix, Access, dBase, archivos .txt, etc.

Al dar un clic sobre el botón  aparecerá una caja de diálogo semejante a la mostrada en la fig. 5.9. Si nosotros elegimos la opción "New" nos aparecerá la siguiente pantalla preguntándonos sobre el tipo de datawindow que deseamos construir:



Donde:

Data Source es la forma en que el datawindow recuperará la información:

Recuperará los datos de una tabla de una base de datos relacional, saltándose los criterios de selección de la cláusula WHERE.

Recuperará los datos de una o más tablas de una base de datos relacional, permitiendo construir criterios de selección de la cláusula WHERE.

Recupera los datos de un archivo SQL (Query).

Usado para recuperar datos de archivo .txt o .dbf.

Los datos son generados por la ejecución de un procedimiento almacenado.

Presentation Style es lo forma en que el datawindows desplegará la información:

Composición de varios estilos de presentación (por ejemplo una gráfica y un reporte).

Referencia cruzada, realiza cálculos cruzando información de una columna X con una columna Y.

Forma Libre, para diseños de pantalla en cualquier formato.

Permite crear diversos tipos de gráficas.

Presentación de los datos en renglones y columnas (**no** permite diseño libre como poner títulos o imágenes).

Creación de reportes con un formato ya predeterminado.

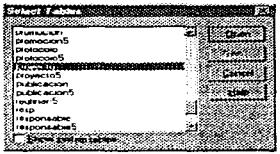
Creación de etiquetas.

Presentación de datos en N columnas, por ejemplo 2 columnas se verían así:

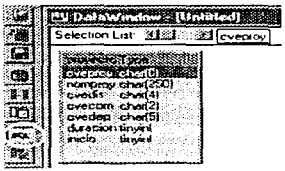
```
IN100296    IN102296
IN103296    IN104296
...
```

Presentación de los datos en renglones y columnas (permite diseño libre como poner títulos o imágenes).

Ejemplo. Realicemos un datawindow usando **SQL Select** y presentación **Tabular**:

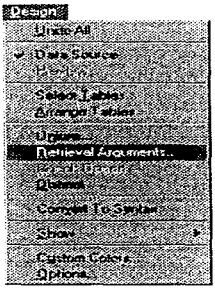


a) Tenemos que elegir las tablas a usar en el datawindow



b) Tenemos que especificar las columnas de las tablas del datawindow

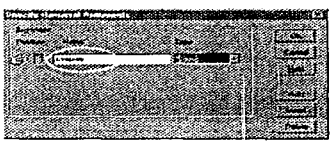
En este caso no vamos a establecer condiciones de selección (no habrá cláusula WHERE).



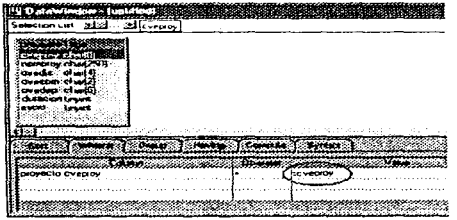
c) *Opcionalmente*, podemos establecer condiciones de selección o filtros **(cláusula WHERE)** de la siguiente manera:

A un datawindow con condiciones de selección lo denominaremos **datawindow con *n* argumentos**, donde n = 1,2,....
Si el datawindow **no** establece condiciones de selección se le **llamará datawindow sin argumentos**.

1. Elegimos Retrieval arguments... del menú Design



2. Especificamos los argumentos y su tipo (string, date, integer, etc.)



3. Establecemos la cláusula WHERE

Los dos puntos en el argumento cveproy (**:cveproy**), indican que se trata de un argumento y no de un valor constante. Los argumentos, así como valores constantes se pueden ver dando clic derecho sobre la columna value (donde dice ":cveproy").

El área donde estamos trabajando ahora se le conoce como el **área de select** (en esta podemos seleccionar otras tablas, relacionar tablas, elegir otros campos, etc.). Si nosotros queremos cambiar la apariencia de nuestro datawindow (agregar textos, pegar imágenes, cambiar colores, dibujar líneas, etc.) necesitamos pasar a un **área de diseño**; esto lo podemos hacer presionando el botón *Select* ( , encerrado en un circulo en la figura del inciso b). En el área de diseño podemos modificar la apariencia de nuestro datawindows, así:

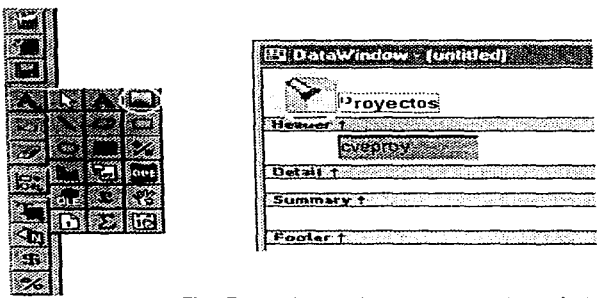
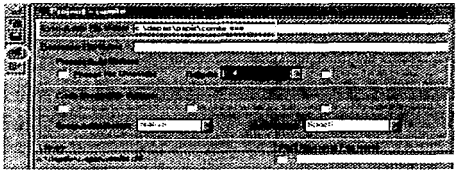


Fig. Pegando una imagen en un datawindow

Con el mismos botón de Select ( ) podemos regresar al **área de select** para seleccionar otras tablas y/o campos.

Crea objetos project (programas ejecutables)

Al dar un clic sobre el botón aparecerá una caja de diálogo semejante a la mostrada en la fig. 5.9. Si elegimos la opción "New" aparecerá la siguiente pantalla:



En donde debemos especificar el nombre del programa ejecutable, así como su destino; después se construye el programa ejecutable oprimiendo el botón de "Build" (encerrado en un circulo en la figura anterior).

El programa ejecutable (comite.exe) podrá correr, independiente del software PowerBuilder, en un ambiente windows.

- **Comunicación interconstruída entre aplicaciones**

Será más sencillo para los programadores de software escribir aplicaciones que puedan intercambiar datos entre sí. Puesto que la interface gráfica entre las GUI's es la misma independientemente del software que se utilice para el desarrollo, el programador puede crear "partes" por separado de una aplicación y después conjuntarlas para obtener una sola; en este sentido,

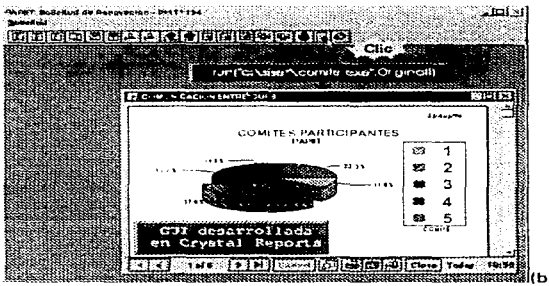
el usuario solamente ve una aplicación sin darse cuenta, y sin importarle, de que se tratan de varias aplicaciones comunicándose entre si.

La comunicación entre GUI's mezcla cualidades de distintas herramientas para el desarrollo de aplicaciones para generar aplicaciones más completas y atractivas. Así, el programador no está limitado al uso de una sola herramienta de desarrollo, sino que, por el contrario, tiene a su disposición varias herramientas.

La tendencia actual de las herramientas de desarrollo es la conectividad de datos, esto es, la comunicación libre entre la aplicación y los datos, sin importar el lugar en donde se encuentren; en este sentido, podemos procesar un conjunto de datos y obtener distintas salidas como pueden ser un reporte, una estadística, una gráfica, etc. y enlazarlas todas en una sola aplicación. Un ejemplo de ello es ODBC (Open DataBase Conenectivity), la cual es una API (Application Programming Interface) desarrollada por Microsoft para proporcionar acceso estandarizado a las aplicaciones para el manejo de diversas fuentes de datos (las cuales son usualmente bases de datos, pero pueden ser otra clase de archivos tales como archivos de texto).

La forma en que se comunican las aplicaciones GUI's es muy simple: un módulo o programa principal es el encargado de ejecutar las aplicaciones GUI (aunque también se pueden ejecutar aplicaciones basadas en caracteres) que fueron desarrolladas en la misma herramienta de desarrollo, o bien, en otra; y una vez que éstas aplicaciones realizan su objetivo (como puede ser el mostrar una gráfica) regresarán el control al programa principal.

(a)



(b)

Fig. La comunicación interconstruida entre GUI's ofrece una grandes ventajas (a) la GUI desarrollada en PowerBuilder tiene como uno de sus objetivos mostrar el comité al que pertenece un proyecto de investigación, (b) la GUI desarrollada en Crystal Reports Professional 4.5 tiene como finalidad mostrar gráficamente los proyectos de investigación existentes por comité. Ambas GUI's usan datos de la misma base de datos.

Con las capacidades de las GUI, los programadores de software tendrán que escribir menos en código fuente y, en consecuencia, se podrán elaborar aplicaciones en un tiempo mucho menor. Asimismo un beneficio más para los programadores de software es la cada día mayor ampliación del mercado de las aplicaciones GUI's, las cuales tienen gran aceptación por su facilidad de uso y por la presentación gráfica que ofrecen.

Dado que las tendencias indican que la GUI representará la interface de usuario por omisión de todas las aplicaciones futuras de mayor aceptación, cualquier aplicación escrita para una GUI tendrá un mercado potencial más amplio Una de las ganancias que se tienen al cambiar a GUI's es que los usuarios de computadoras pueden aprenderse más rápido y con el mejor provecho, aumentando con esto su productividad. La justificación de esto es sencilla, un usuario de computadoras que utiliza una aplicación GUI encuentra similitud entre las aplicaciones, por lo que ya no se le tiene que enseñar ciertas cosas, el retoma lo aprendido en otras aplicaciones y, en consecuencia, ahorra tiempo en el aprendizaje.
que una aplicación no GUI.

Visualizadores

El concepto de herramienta mecanizada que brinde soporte al conocimiento es una idea antigua, en 1945 Vaner Bush propone el sistema " Memex" en donde su idea fundamental era que la selección de la información se realizara por asociación en lugar de por indexación, esto como posible solución en el manejo de grandes cantidades de información. Para 1962 Douglas Engelbart trata de definir las funciones que se deberían incorporar a las computadoras para aumentar las capacidades humanas, entre estas destacan conexiones entre textos, correo electrónico, librerías, documentos y espacios privados para uso personal.

Vaner Bush y Engelbart creían en la habilidad de una máquina para aumentar la inteligencia del ser humano y comunicar esa inteligencia entre ellos. Vaner Bush predijo la explosión de la información científica y la necesidad de una máquina para ayudar a los científicos a continuar con el desarrollo de sus disciplinas por medio de consultas a documentos sin la necesidad de tener un seguimiento lineal, sino por el contrario, tener acceso a la información de más interés sin necesidad de hacer búsquedas en una cantidad ilimitada de información almacenada en grandes bancos de datos.

Para 1965 Theodor Nelson tomando como antecedentes las ideas de Vaner Bush y Engelbart inventa el término ***hipertexto*** que es básicamente lo mismo que texto común, en el sentido de que puede ser leído, guardado y editado, de manera estricta se aplica a los sistemas basados únicamente en texto.

Hipertexto: No es más que un documento o texto en el cual determinadas palabras o frases derivan a un nuevo documento relacionado con el contenido de dicha palabra o frase.

Los aspectos más destacados de estos sistemas son la posibilidad de soportar múltiples usuarios y múltiples ventanas además de añadir un periférico al teclado ordinario, el ratón, que actualmente incorporan la mayor parte de los sistemas.

Así mismo la preocupación de que los usuarios no se vieran envueltos en un mundo de teclas y comandos difíciles aparecieron en Europa en 1992, los primeros ***vizualizadores o "browsers"***, pero no es sino hasta 1993 cuando aparece por primera vez el programa Mosaic que constaba de progra-

mas basados en texto que con simples comandos del teclado brindaba gran facilidad en el manejo de la información a los usuarios.

Visualizador: Programa interactivo usado para acceder a la información del WWW.

Visualizador o browser

Un vizualizador o browser es un software que interpreta documentos en hipertexto, permite acceder a la información disponible representándola en un conjunto de elementos para el acceso a grandes bases de datos de texto e imágenes en diversas máquinas.

Los documentos en hipertexto son conocidos como páginas que constan de diferentes elementos que permiten la transmisión de información y su acceso de un modo sencillo. Dependiendo de que si la información este o no bien estructurada se encontrara lo que estamos buscando de una manera fácil. Es decir el término hipertexto sólo se aplica en sistemas basados en texto, ***hipermedia*** por el contrario es una extensión del hipertexto para incluir datos ***multimedia*** (imágenes, audio, video). Aquí los enlaces son conexiones visuales gráficas o fotográficas, mensajes de audio video o texto.

Hipermedia: No es si no una extensión del hipertexto en el que se incluyen datos multimedia.

Hipertexto

El hipertexto permite a los usuarios desplazarse de un documento a otro de una manera no secuencial ni lineal. Las palabras, frases e iconos del documento se convierten en enlaces o ligas que permiten viajar a una nueva posición en el mismo documento o incluso a uno nuevo. Así es, estas ligas o enlaces nos permiten el acceso inmediato a otros documentos que nos proporcionarán mayor información relacionada con el documento original. Podríamos ver esto como el equivalente a una nota de pie de página, o a una referencia bibliográfica, pero con la ventaja de que la computadora nos da acceso inmediato a esta nueva información.

Multimedia: significa "Multiples medios", es decir que la información obtenida por el WWW puede ser desde un archivo texto o gráfico, hasta archivos de sonido, video y animaciones.

El manejo de hipertexto presenta algunas ventajas, como son:

- Cada uno de los enlaces puede representar un documento, un índice o el resultado de una búsqueda indexada
- Facilita el manejo en documentos muy largos
- Proporciona velocidad de uso
- Brinda profundidad en sus búsquedas

En cuanto a los sistemas de hipermedia, estos deben contar con los siguientes criterios:

- Un medio ambiente adaptativo para la integración de herramientas de datos que no confinen al usuario a editores en particular ni a paquetes especiales de software
- Un sistema cuya plataforma sea independiente de las demás y que pueda distribuir documentos entre el resto de las plataformas
- Un sistema que facilite al usuario encontrar, actualizar e intercambiar información
- Y por último un sistema en el cual todas los formas de los datos y medios es tratado conceptualmente de una forma similar

La hipermedia le da vida al documento y la computadora se convierte en un dispositivo multimedia.

Multimedia es cualquier combinación de texto, gráfico, sonido, animación y video que llega a nosotros por computadora. Cuando uno permite a un usuario final controlar ciertos elementos y cuando deben presentarse, se le llama multimedia interactiva. Cuando uno proporciona una estructura de elementos ligados a través de los cuales el usuario puede navegar, entonces se convierte en Hipermedia.

Los elementos de multimedia se conjugan en un proyecto utilizando herramientas de desarrollo de multimedia. Estas herramientas de programación están diseñadas para administrar los elementos de multimedia individuales y permitir que interactúen los usuarios. La mayoría de las herramientas de desarrollo de multimedia ofrecen facilidades para crear y editar texto e imágenes, y tienen extensiones para controlar los reproductores de videodiscos, videos y otros periféricos relacionados.

El equipo y los programas que rigen los límites de lo que puede ocurrir es la plataforma o ambiente multimedia.

Es conveniente utilizar multimedia cuando las personas necesitan tener acceso a información electrónica de cualquier tipo. Multimedia mejora las interfaces tradicionales basadas sólo en texto y proporciona beneficios importantes que atraen y mantienen la atención y el interés del usuario. Multimedia mejora la retención de la información presentada.

Mosaic: Visualizador de la WWW

Visualizadores más populares en el Web

Los visualizadores del ***Web*** mas populares son los siguientes:

- NCSA Mosaic
- Netscape

WWW (World Wide Web): Servidor de información, desarrollado en el CERN(Laboratorio Europeo de Física de Partículas), buscando construir un sistema distribuido hipermedia e hipertexto.

NCSA Mosaic fue diseñado como un vizualizador de la ***World Wide Web***, la primera versión surge de manera formal en 1993 y fue diseñada por Marc Andersen y Eric Bina en NCSA específicamente para el sistema X-Window y para septiembre de ese año hacen su aparición las primeras ***versiones beta*** para Macintosh y Windows de Microsoft, presenta una interfaz multimedia a Internet. Mosaic se basa en un modelo cliente/servidor en el manejo de la información, esta basado en la tecnología World Wide Web de CERN haciendo uso de las herramientas (bibliotecas) proporcionadas por los clientes de WWW.

Versiones beta: Se refieren al software no tiene costo alguno y su fin es de evaluación.

Realiza muchas más tareas que la simple presentación de hipertexto con vínculos hacia otros documentos; se trata de una herramienta hipermedia, lo cual significa que puede manejar audio, imágenes e incluso video y significa la interfaz
con diferentes servicios.

Mosaic es un ejemplo de cliente WWW para sistemas basados en ventanas. Está configurado para acceder directamente desde los menús de windows a servicios como WAIS, Archie, Whois, Techinfo, HyTelnet, X.500, finger y otros muchos más.

Lleva implementados los protocolos ***HTTP***, Gopher, FTP y NNTP, y el acceso a WAIS, Techinfo, Archie, finger, X.500 y Whois es a través de pasarelas. Está disponible en la mayoría de los servidores FTP en Internet.

HTTP: Protocolo de Transferencia de Hipertexto (HyperText Transfer Protocol). Es un protocolo (conjunto de reglas y normas) que se usa para definir las comunicaciones entre programas clientes y servidores WWW

Para poner en funcionamiento el Mosaic es muy sencillo, basta con teclear desde el prompt del sistema operativo:

%Mosaic

o dar doble clic desde el icono de Mosaic estando en un ambiente gráfico y se presentará la una pantalla como la que se muestra en la figura 6.2.

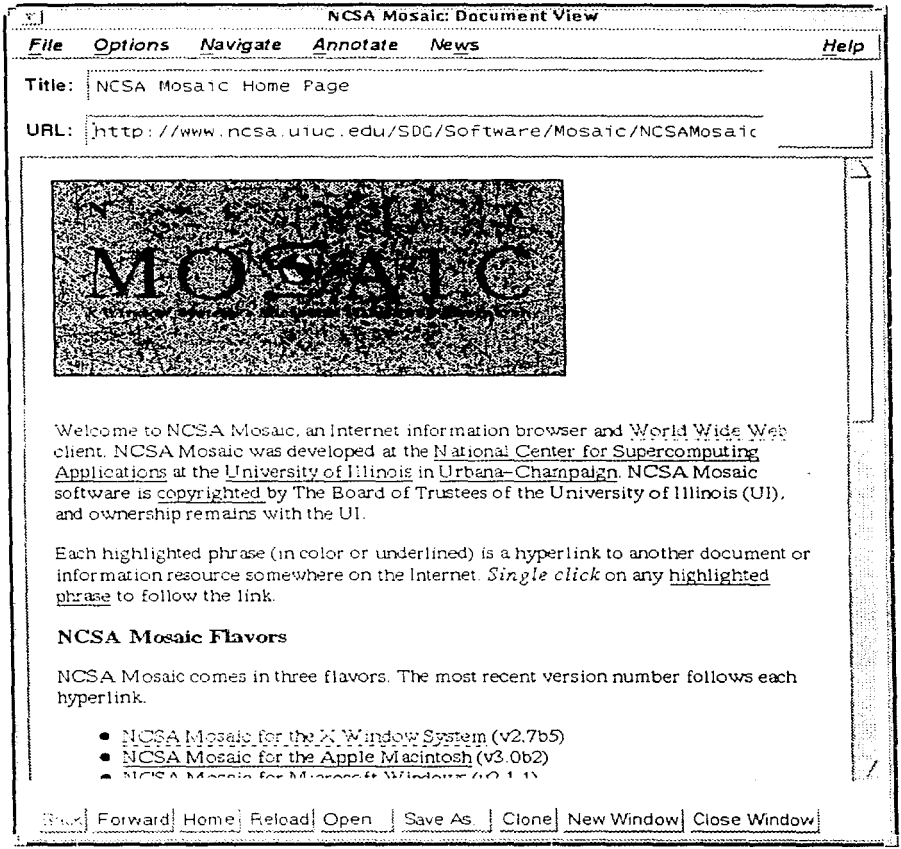


Fig. 6.2 Pantalla Mosaic

Netscape es un instrumento de software que nos permite interactuar en un ambiente gráfico en Internet. Básicamente lo podemos definir como una interfase que permite la comunicación de las computadoras en las redes mundiales de Internet

Netscape: Vizualizador de WWW.

Esta capacidad nos permite hacer la transmisión de textos e imágenes (inclusive sonidos) a través de las líneas de las redes de cómputo.

La forma en la que se opera Netscape es sumamente sencilla y amigable para el usuario. Es suficiente con que se coloque el cursor del ratón sobre cualesquiera de los textos que aparecen en color y subrayado, para que la máquina efectúe una conexión a otro documento, marcado por ese indicador. Estos indicadores reciben el nombre de ***"links" o ligas*** (uniones). Es interesante destacar que cuando presionamos el ratón sobre una liga y se efectúa la conexión a otro documento, este documento puede estar físicamente en la misma computadora en la que trabajamos, en otra computadora dentro del mismo centro de computo, o en cualquier otra computadora de otra localidad, de otro país o en cualquier punto del mundo.

Link: salto de una página a otra. Se utiliza para pasar de una página a otra, ya sea que esté en el servidor local o en algún servidor en otra parte del mundo. Así es como se provee el mecanismo de navegación.

Netscape proporciona la "entrada" a un número inmenso de fuentes de información los países del mundo.

Para poner en funcionamiento el programa de Netscape es muy sencillo, basta con teclear desde el prompt del sistema operativo:

%Netscape

- dar doble "clic" sobre el icono de Netscape estando en un ambiente gráfico y se presentará la una pantalla como la que se muestra en la figura 6.4.

Navegación: buscar/mirar información a través de las pági-nas WWW saltando (mediante los links) de un lugar a otro sin que el usuario sepa necesariamente , el lugar del mundo en que se encuentra.

Fig 6.4 Pantalla de Netscape

❖ HyperText Markup Language (HTML)

HTML (Hypertext Markup Language) Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986(SGML,Standard Generalized Markup Language).

Los autores de hipermedia utilizan un lenguaje hipertexto conocido como Hypertext Markup Language (***HTML***), que es parte de la tecnología que hace posible este modo de acceder a la información; consiste en una serie de definiciones estándar que le indican a la página si el contenido es texto, gráfica, sonido o video, este lenguaje puede representar hipermedia, menús, resultados de Base de datos y algunas otros formatos de información.

HTML fue especificado por primera vez en 1991 por Tim Berners Lee, actualmente es considerado el motor central de lo que hoy se conoce como World Wide Web y que tanta atención ha ganado en los últimos tiempos. El lenguaje HTML no tenía contemplado el despliegue en pantalla en todo su potencial ya que el usuario tenía una interfaz limitada a un mundo de texto en blanco y negro, y ligas que permitían visualizar gráficas. Los primeros pasos estaban dados, y decenas de organizaciones comenzaron a construir páginas que cumplieran con el estándar HTML y las ponían a disposición de cualquier usuario de Internet para su consulta sin costo alguno.

Este lenguaje de marcado, nos permite definir la estructura lógica del documento, más que la definición de un formato específico, con tipos y tamaños de letra, número y tamaño de líneas, etc. De esta forma, el documento puede ser desplegado en distintos sistemas de acuerdo a las posibilidades de cada uno de éstos. La apariencia final con la que aparecerá el documento, depende, en última instancia, de las características del ***programa cliente***.

Programa Cliente: Programa que utiliza recursos de otra máquina

Con HTML podemos definir, entre otras cosas, el principio y final de cada párrafo, distintos tipos de encabezado, resaltar textos con cursivas o negritas, crear listas de distintos tipos, incorporar imágenes y claro está, definir las ligas a otros documentos.

ASCII :Conjunto de ca-racteres sin formato alguno.

Un documento en HTML es un archivo ***ASCII*** donde cada uno de los elementos o estructuras del mismo se marca utilizando etiquetas.

La mayoría de los comandos HTML se presentan en parejas. Una que

inicia el modo de presentación (<>) y otra que lo finaliza (</>).

Un documento HTML comienza con la marca <HTML> y termina con la marca </HTML> , este se compone de dos partes "cabeza" <HEAD>...</HEAD> y "cuerpo" <BODY>...</BODY>.

En la cabeza suele escribirse el título que aparecerá en la parte superior del vizualizador que estemos utilizando, en el cuerpo el contenido de la página.

La estructura básica de un documento en HTML es la siguiente:

```
<HTML>
<HEAD>
<TITLE>PAPIIT: Renovación </TITLE>
</HEAD>
<BODY>
<H1> Dirección General de Asuntos del Personal Académico</H1>
</BODY>
</HTML>
```

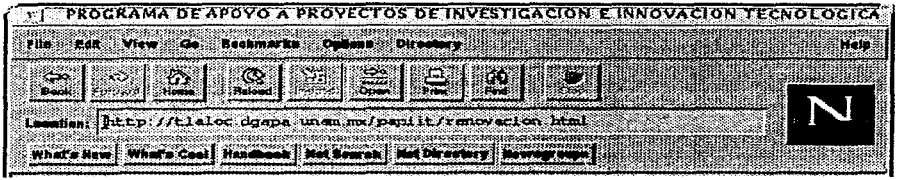
 Los comandos expresados en las marcas o etiquetas no son sensitivos, esto es, que pueden ser escritos en letras mayúsculas o minúsculas (<HTML> es lo mismo que <html>).

A continuación se explica la función y uso de las marcas o etiquetas más usadas en HTML.

- **TITLE (Título):**

Esta etiqueta sirve para poner un título en la barra del vizualizador. El texto va entre las etiquetas **<TITLE></TITLE>** y debe ponerse en la cabeza del documento, es decir entre las etiquetas **<HEAD></HEAD>.**

```
<HEAD>
<TITLE> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN TECNOLÓGICA</TITLE>
</HEAD>
```

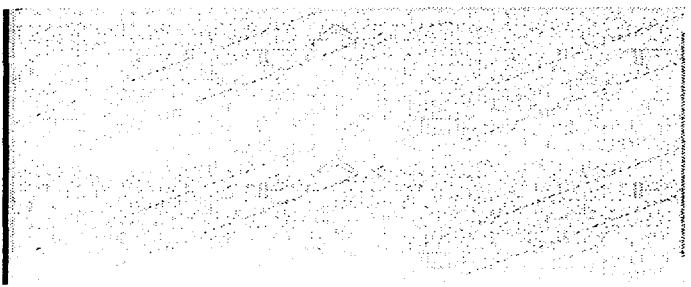


- **BACKGROUND (Papel Tapiz):**

El papel tapiz del documento es el fondo de la página, se pone dentro de las etiquetas <BODY></BODY> :

Es necesario indicar la ruta completa de donde se extraerá el papel tapiz, indicando la dirección del servidor y el nombre del archivo gráfico.

```
<BODY BACKGROUND="http://tlaloc.dgapa.unam.mx/imagenes/bk_dgapa.gif" >
.
.
.
</BODY>
```

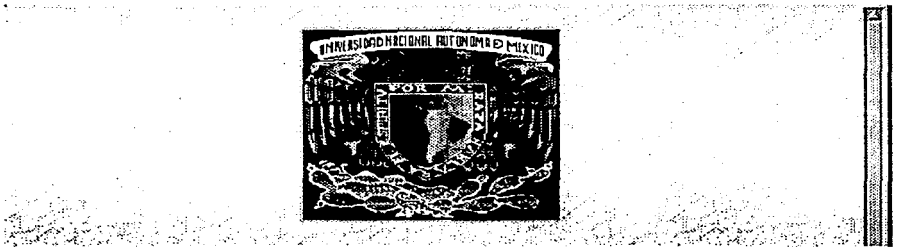


- **IMG (Imagen):**

Es una imagen que puede ser insertada en la página, el formato de la imagen debe ser .gif o .jpg para vizualizadores Netscape y .gif para vizualizado

res Netscape y gif para vizualizadores Mosaic. La inserción de la imagen se realiza mediante la instrucción <IMG SCR="nombre_imagen.extensión" ALING=LEFT|RIGHT| CENTER>.

```
<IMG SRC="escudo.gif"  ALING=CENTER>
```



- **CENTER (Centrado):**

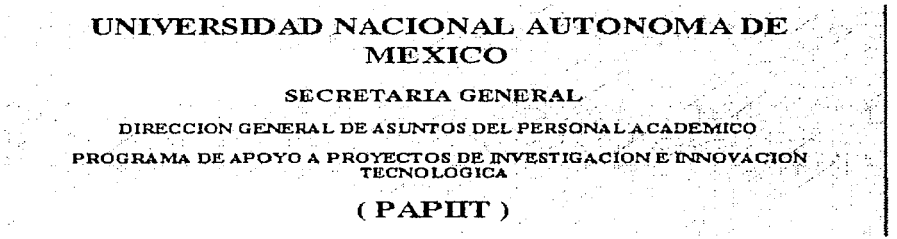
Este comando sirve para centrar imágenes, textos o líneas. Se deben colocar las etiquetas <CENTER></CENTER>. Por ejemplo las siguientes instrucciones nos muestran el mismo resulta del ejemplo anterior.

```
<CENTER>
<IMG SRC="escudo.gif">
</CENTER>
```

- **H1, H2, H3, H4, H5 (Tamaño de letra):**

El texto se puede poner en diversos tamaños mediante las etiquetas**<H1></H1>, <H2></H2>, <H3></H3>, <H4></H4>, <H5></H5>**. El tamaño del texto será proporcional al número que acompañe a la etiqueta como se muestra a continuación.

```
<CENTER>
<H1>UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO </H1>
<H3>SECRETARIA GENERAL</H3>
<H4>DIRECCION GENERAL DE ASUNTOS DEL PERSONAL ACADEMICO</H4>
<H4>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN
TECNOLOGICA</H4>
<H2>(PAPIIT)</H2>
</CENTER>
```



- **LINK (Liga o enlace):**

Las ligas nos permiten trasladarnos de un documento a otro de una manera no secuencial ni lineal, haciendo referencia a otra página residente en el mismo o en otro servidor. Se utilizan las etiquetas: <A HREF> </A>

```
< A HREF="http://tlaloc.dgapa.unam.mx/papiit/Inicio_ren.html>SOLICITUD DE
RENOVACION 1996-1997 </A>
```

- **TABLE (Tabla):**

Las tablas se utilizan para agrupar la información en celdas. Utiliza las etiquetas <TABLE WIDTH=ancho_de_la_tabla% BORDER=tipo></TABLE>. El ancho de la tabla se refiere al tamaño de la misma a lo ancho de la pantalla, este debe especificarse en porcentaje tomando como referencia el

ancho de la pantalla como el 100%, el tipo borde es el marco de la tabla y el valor que puede tomar es un número del 1 al 7 refiriéndose específicamente al ancho del borde; si se elige el número cero el borde no será visible. Para definir un de renglón se utilizan las etiquetas <TR></TR> y para definir una columna <TH></TH>.

```
<TABLE><TR><TH>
<A HREF="http://tlaloc.dgapa.unam.mx/papiit/Inicio_ren.html">
SOLICITUD DE RENOVACION 1996-1997</A>
</TH></TR></TABLE>
```

El resultado de este ejemplo se puede observar viendo el recuadro en la figura del ejemplo anterior.

- **BLINK (Texto parpadeante):**

Cuando se desea que un párrafo, línea, palabra o imagen parpadee, se usan las etiquetas **<BLINK></BLINK>**.

```
<BLINK> IMPORTANTE: </BLINK>
```

IMPORTANTE:

- **PRE, P, BR (Etiquetas de edición):**

Dado que HTML ignora los espacios en blanco es necesario utilizar las etiquetas <PRE> </PRE> para justificar un texto si esta requiere espaciado.

La etiqueta <P> nos sirve para escribir un texto en una nueva línea dejando un espacio entre línea y línea.

La etiqueta
 nos sirve para escribir un texto en una nueva línea sin dejar espacio entre línea y línea.

Tanto <P> como
 son etiquetas únicas y no requieren de una etiqueta para su finalización.

```
<PRE>
La Dirección General de Asuntos del Personal Académico,
comunica a los investigadores y profesores titulares de
carrera de tiempo completo que registraron proyectos en el
Programa de Apoyo a Proyectos de Investigación e Innovación
Tecnológica, que las fechas límite de recepción de
solicitudes de renovación al Programa son las siguientes:

Ciencias Físico-Matemáticas y de las Ingenierías  4 de junio
Ciencias Biológicas y de la salud          5 y 6 de junio
Ciencias Sociales                          3 de junio
Humanidades y Artes                        3 de junio
Innovación Tecnológica                     3 de junio

Por lo tanto, a partir de las fechas señaladas se cerrará el
sistema disponible en WWW de acuerdo a cada comité.
Asimismo, después de estas fechas ya no podrán realizar
cambios en la información capturada.
MAYO, de 1996. </PRE>
```

- **B, I, U (Formato de Texto: Negrita, Itálica, Subrayado):**

Cuando se desea poner diferentes estilos de letras se pueden usar las siguientes etiquetas:

<B></B>	Tipo Bold	**PAPIIT**
<I></I>	Tipo Itálica	*PAPIIT*
<U> </U>	Subrayado	PAPIIT

Conjuntando los ejemplos anteriores obtendremos el siguiente archivo HTML :

```
<HTML>
<HEAD>
<TITLE>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACION E INNOVACION
TECNOLOGICA</TITLE>
</HEAD>
<BODY BACKGROUND="http://tlaloc.dgapa.unam.mx/imagenes/bk_dgapa.gif">
<CENTER>
<IMG  SRC="escudo.gif" ALT=" imagen DE APOYO A PROYECTOS
 DE INVESTIGACION E INOVACION TECNOLOGICA"></IMG>
</CENTER>
<CENTER><H2>UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO</H2>
<H3> SECRETARIA GENERAL </H3>
<H4> DIRECCION GENERAL DE ASUNTOS DEL PERSONAL ACADEMICO</H4>
```

```
<H4> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACION E INNOVACION
TECNOLOGICA</H4>
<BLINK><H2>( PAPIIT ) </H2></BLINK>
<FONT SIZE=7>
<CENTER>
<PRE>
<TABLE><TR><TH>
<A HREF="http://tlaloc.dgapa.unam.mx/papiit/Inicio_ren.html">

SOLICITUD DE RENOVACION 1996-1997</A>
</TH></TR></TABLE>
</PRE>
</FONT></PRE></CENTER></FONT>
<BLINK><H3>IMPORTANTE:</H3></BLINK>
<B>HORARIO DE SERVICIO DEL SISTEMA: Las 24 horas del día durante toda la semana
excepto los días Lunes a Viernes de
9 a 10 de la mañana por respaldo de información. <p>
<PRE>
La Dirección General de Asuntos del Personal Académico,
comunica a los investigadores y profesores titulares de
carrera de tiempo completo que registraron proyectos en el
Programa de Apoyo a Proyectos de Investigación e Innovación
Tecnológica, que las fechas límite de recepción de
solicitudes de renovación al Programa son las siguientes:

Ciencias Físico-Matemáticas y de las Ingenierías  4 de junio
Ciencias Biológicas y de la salud          5 y 6 de junio
Ciencias Sociales                        3 de junio
Humanidades y Artes                        3 de junio
Innovación Tecnológica                     3 de junio

Por lo tanto, a partir de las fechas señaladas se cerrará el
sistema disponible en WWW de acuerdo a cada comité.
Asimismo, después de estas fechas ya no podrán realizar
cambios en la información capturada.
MAYO, de 1996. </PRE>
</BODY>
</HTML>
```

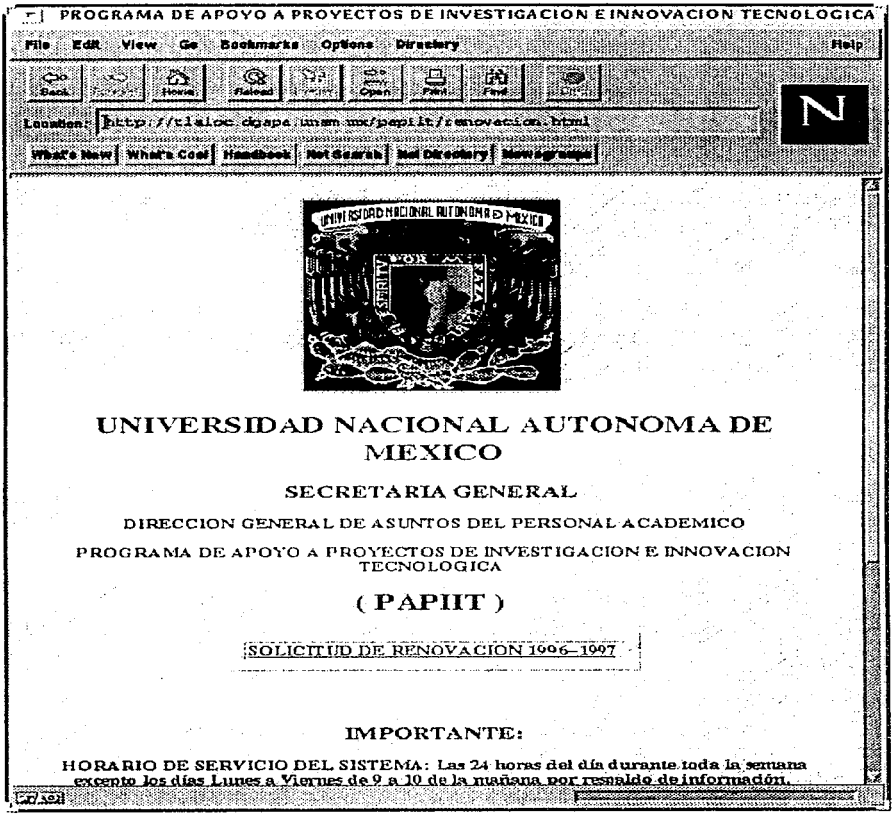


Fig 6.1 Programa en HTML

◆ **FORM (Formas)**

HTML provee una manera práctica para el envío de información de páginas Web al servidor de Internet (HTTP). Esto se realiza a través de elementos llamados formas, representados por las etiquetas **<FORM> </FORM>.** En el capítulo 7 siguiente veremos más a detalle las formas.

Las formas soportan diferentes tipos de objetos, como son los siguientes:

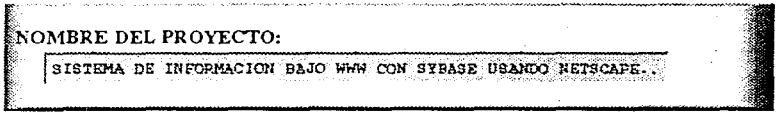
- TEXT
- PASSWORD
- CHECKBOX
- RADIO
- SUBMIT
- RESET
- HIDDEN
- SELECT
- TEXTAREA
- TABLE

La estructura general de los objetos mencionados anteriormente es la siguiente:

<INPUT TYPE="objeto" NAME="nombre_del_objeto" VALUE="valor_del_ objeto">

A continuación se explicará cada uno de estos objetos:

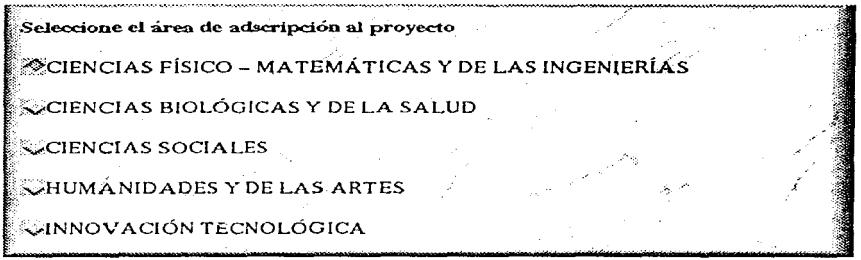
TEXT.- Campo de entrada de texto, es el default.



PASSWORD.- Campo de entrada de texto, la entrada es representada por asteriscos.

CHECKBOX.- Un simple botón de switch (activo/no activo).

RADIO.- Un simple botón de switch (activo/no activo); otros botones con el mismo nombre son agrupados dentro de uno de muchos, es decir son mutuamente excluyentes.



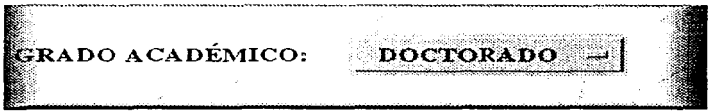
SUBMIT.- Un pulzador que causa que la forma actual sea empacada dentro de la consulta "URL" y enviada al servidor remoto.

RESET.- Un pulzador que causa que los valores contenidos en la forma, sean reinicializados a sus valores por default.

HIDDEN.- Palabra clave que especifica que el campo es oculto.

MENÚS.- Dentro de este tipo de menús nosotros podemos fijar el tamaño ya que aveces se presentan listas muy grandes de información, para estos casos se anexará el código que servirá para especificar el número de renglones que serán visualizados en pantalla, esto dentro de la cláusula <select> de la siguiente forma:

Este código lo observaríamos en pantalla como se muestra en la siguiente figura:




```
<h4>GRADO ACADEMICO:<select name=DOCTORADO SIZE=N>
```

Así también existen objetos para información muy extensa como es el TEXTAREA, cuyo formato general se presenta a continuación.

```
Ejemplo
<textarea name=conclusion rows=20 cols=71>%s,objetivo
</textarea><p>
```

Lo cual será observado en pantalla como se muestra a continuación:

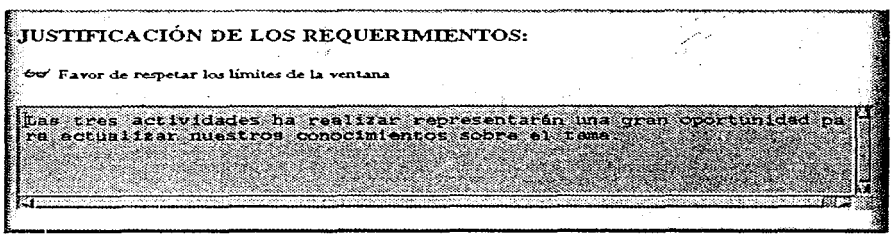


TABLE (Tablas):

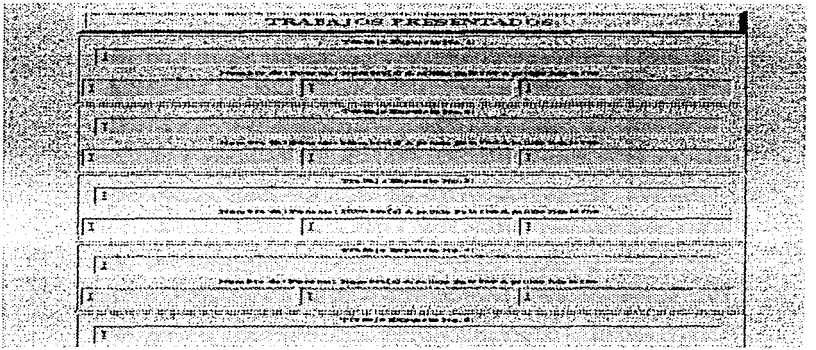
Las tablas nos permiten agrupar información. Se utilizan las etiquetas <TABLE></TABLE>.

Una tabla esta conformada de columnas indicándolas mediante <TD></TD> y renglones <TR></TR>, así como también podemos manejar diferentes tipos de bordes para una tabla lo cual se indicar dentro de la etiqueta de TABLE con la palabra BORDER igualando a un número que puede ir desde 1 hasta 5 y dependiendo del numero será en grosor del BORDER.

Cuando en una tabla deseamos separar una columna en más columnas se realizara mediante la etiqueta <COLSPAN = #> , en donde # indica el número de columnas que deseamos

A continuación se presenta el código de una tabla.

```
<CENTER><TABLE WIDTH=90%% BORDER=5>
<TR><TH ALIGN=CENTER COLSPAN=2><FONT SIZE=+1
COLOR=FF0000>TRABAJOS PRESENTADOS</FONT></TH></TR>
</TABLE></CENTER>
<CENTER><TABLE WIDTH=90%% BORDER=1><TR>
<TH><FONT SIZE=-2>Trabajo Expuesto No. %d:
</FONT><BR>
<INPUT NAME=trab TYPE=TEXT size=63 MAXLENGTH=100>
<BR><FONT SIZE=2>
Nombre del Ponente: Nombre(s)
Apellido Paterno Apellido Materno</FONT><BR>
<INPUT NAME=nombre TYPE=TEXT size=20 MAXLENGTH=20>
<INPUT NAME=apellidop TYPE=TEXT size=20 MAXLENGTH=20>
<INPUT NAME=apellidom TYPE=TEXT size=20 MAXLENGTH=20>
</TH>
</TR>
</TABLE></CENTER>
</TABLE>
```



HTML sigue un modelo de desarrollo abierto. Cuando una nueva característica es propuesta, es implementada en algunos clientes y probada en algunas aplicaciones. Si la demanda para esta nueva característica es suficiente, otras implementaciones son animadas a seguir esta nueva demanda y la nueva característica llega a ser ampliamente empleada. En este proceso, el diseño es revisado y quizás modificado o potenciado. Finalmente, cuando existe suficiente experiencia con esta nueva característica, llega a ser parte del conjunto estándar de HTML.

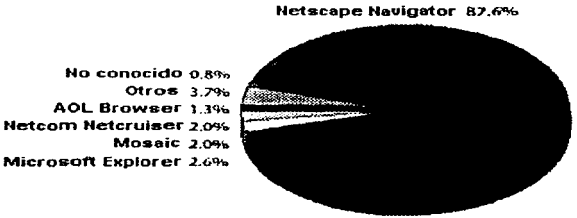
✓ Porqué usar Netscape?

El Netscape comparado con el Mosaic es más rápido , brinda facilidad de uso e instalación ya que solo hace falta con entrar a la pagina de Netscape y bajar el software.

Además puede leer los formatos gif y jpg, no como el MOSAIC que solo lee imágenes en formato gif y esto brinda algunas ventajas ya que :

El formato jpg soporta mayor número de colores, es más comprimido, ocupa 3 o 4 veces menos espacio de disco que el gif lo que resulta importante por el espacio.

Según estudios los navegadores están repartidos de la siguiente manera: Netscape ocupa un 87.6 %, Microsoft un 2.69%, NCSA Mosaic un 2.09%, Netcom Netcruiser 2.09%, AOL browser 1.3 %, Otros 3.7% y No conocidos 0.8%.



Fuente IDC's Global Internet User Survey, Julio 1998

Servidores HTTP

La alternativa Cliente/Servidor nos invita a pensar en una amplia gama de servicios, donde, obviamente, el servidor proporciona el servicio para el que fue creado, mientras que el cliente manipula y presenta los datos regresados por el servidor de la manera que más convenga a una aplicación. Una caso particular de este tipo de servicios y que esta siendo abundantemente demandado, es World Wide Web (WWW). Como ya sabemos, WWW utiliza como ***protocolo de comunicación*** el HyperText Transfer Protocol (***HTTP***). En un inicio HTTP (versión 0.9) era un protocolo para el intercambio de texto (llano o hipertexto), posteriormente, con NCSA se extiende su uso para permitir la transmisión y despliegue de archivos en ***formato GIF***. Rápidamente se observó que el Web era útil para la transmisión de prácticamente todo tipo de formatos y aunque capaz de transferir esta multitud de formatos, el que fundamentalmente utiliza es HTML, que describe la estructura interna de cada documento y las ligas de hipertexto a las que apunta. Una liga (o URI, que explicaremos mas adelante) apunta a la dirección de algún recurso de Internet, ya sea otra página en HTML, un archivo GIF, una página del gopher, un archivo en un sitio de ftp, etc.; mientras que con un URL (que se describirá más tarde) indica cómo acceder a dicho documento. A pesar de su versatilidad, este protocolo es sorprendentemente simple.

Típicamente, un cliente de WWW, es un navegador de Web (o ***visualizador***) tal como Mosaic. Netscape, el cual inicia la transacción con una requisición al servidor en la computadora donde se encuentra el recurso a acceder. Esta transacción consiste en cuatro pasos:

1. Establecer la conexión. El cliente establece una conexión al sistema servidor de Web. Esta conexión opera, por omisión, en el ***puerto*** 80 de ***TCP/IP***. El servidor de Web se encuentra en escucha permanente de dicho puerto y en cuanto establece la conexión espera una requisición del cliente.


Protocolo de comunicación: Conjunto de normas que regu lan la recepción y la transmisión de datos.


HTTP: Es un protocolo de transferencia de hipertexto.


GIF(Graphic Interchange Format): Formato de archivos para manipulación de gráficos.


Visualizador: Software que nos sirve para visuali zar documentos en el Web.


Puerto: Canal de comunicación, en este caso hace referencia a un puerto lógico (no físico).


TCP/IP (Transmision Control Protocol /Internet Protocol): Conjunto de protocolos de comunicaciones

2. Requisición. Un cliente puede efectuar dos tipos básicos de requisiciones:

Línea de requisición: Es una línea de comando que realiza peticiones al servidor.

Return: La tecla más grande del teclado de una computadora, que finaliza la entrada de un comando o una línea en un archivo de texto.

LineFeed: Código de caracteres que mueve el cursor de la pantalla o de la impresora a la línea siguiente.

a) Requisición simple. Una requisición simple (simple request) obtiene un documento en HTML o cualquier otro formato. La ***línea de requisición*** tiene la forma:

GET <URL> <return>

donde el <return> es un ***Return*** junto con un ***LineFeed***. El servidor siempre responde a esta requisición con un documento. Si hay algún error, por ejemplo, que el archivo indicado por la URL no exista en el servidor, éste reporta al cliente un mensaje de error en inglés en formato HTML. Un ejemplo de como funciona el protocolo en este tipo de requisición, es la siguiente sesión de transmisión manual de un documento por el Web, esta sesión tiene como objetivo recuperar el documento /www/papiit/rfc.html.

```
$telnet tlaloc.dgapa.unam.mx 80
GET /www/papiit/rfc.html
<HTML>
<HEAD><TITLE>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
INNOVACIÓN TECNOLÓGICA</TITLE></HEAD>
<BODY>
Proporcione su RFC:
</BODY>
Conecction Closed by foreign host.
$
```

a) *Requisición extendida*. Permite el uso de modificadores al comando GET arriba mencionado. El formato del comando es

<Método> <URI> HTTP/1.0 <return> [Modificadores *] [<return> <datos>]

Caché: Es parte del disco duro de una computadora que se utiliza como memoria para alojar archivos, con la finalidad de ahorrar tiempo al leerlos nuevamente.

Un ejemplo, es el modificador "if-modified-since" con el cual se puede evitar bajar un archivo que se encuentra en ***caché*** local si éste no ha sido modificado. El '*' indica que es posible tener más de un modificador de comando por requisición.

3. Respuesta. Las respuestas a requisiciones sencillas son documentos o archivos. Mientras que la respuesta de una requisición extendida es un mensaje de la forma

HTTP/1.0 <número clave> <razón> <return> <??Data ??>

El 'número clave' indica si la requisición pudo ser efectuada exitosamente o si hubo algún error y de qué tipo.

4. Cerrar la conexión. Se establece un nueva conexión cada vez que se requiere de un archivo y se cierra esta una vez que se envía una respuesta. Esto es, por ejemplo, si una página contiene texto y tres imágenes genera cuatro conexiones cuando es leída, una para el texto y una para cada una de las imágenes. Como vemos, las conexiones no son exclusivas, ni permanentes, por lo que el servidor puede atender más de una requisición al mismo tiempo.

Nótese que la conexión es cerrada automáticamente por el servidor justo después de la transmisión del documento. Se pretende que nuevas versiones de HTTP permitan establecer conexiones extendidas que se mantengan abiertas para más de una transacción y donde el cliente indique al servidor con un modificador de comando que mantenga la conexión hasta

que se le pida que la cierre.

Uniform Resource Identificator

Para continuar hablando del funcionamiento del protocolo HTTP, necesitamos mencionar a los identificadores que proporciona para el uso de recursos. Los URI/URL/URN son la abreviatura de Uniform Resource Identificator/Locator/Name respectivamente. Los URL y URN son tipos de URI. El comando para hacer uso de estos identificadores es:

```
nombre_de_protocolo://identificador_único
```

donde el nombre_de_protocolo puede ser http, ftp, nntp, afs, news y mailto entre otros.

Los URL's son identificadores que consisten del nombre de la máquina (por omisión es la máquina local) y del nombre del archivo, programa o recurso a ser accedido, por ejemplo

```
http://tlaloc.dgapa.unam.mx/papiit/solicitud.html
```

Por otra parte los URN, son identificadores que actualmente no han sido implantados, pero que serán nombres asignados a recursos (computadoras, archivos, procesos, dispositivos, etc.) que se transformarán a una dirección física por un Domain Name Server tal como sucede con los servidores de nombres.

Si alguna vez hemos utilizado el Web, nos percataremos de que tiende a ser muy lento, sobre todo a través de líneas telefónicas, incluso en archivos pequeños tipo texto que aparentemente deberían ser de rápido acceso.

El cuello de botella podría ser la capacidad de transmisión del módem. Pero para el acceso de archivos pequeños de texto, el cuello de botella está dado por el establecimiento de la conexión misma, más que la transmisión de la información. Pues, para establecer la conexión más simple, se requiere enviar un mínimo de siete ***datagramas***, y si a esto le agregamos que estamos accediendo un servidor que intenta identificar el nombre del usuario que solicitó la información (en el cliente), resulta que el tiempo para establecer la conexión, al menos, se duplica. Y si además, incluimos que el tiempo de respuesta que puede ser de hasta 4 segundos antes de que el archivo en cuestión pueda comenzar a ser enviado. Tenemos una servicio que consume en un momento dado una cantidad de tiempo considerable.

Datagrama: Grupo contiguo de datos, que sirven como unidad de mensaje TCP/IP que contiene las direcciones de origen y destino de la Internet y los datos.

Servidor y cliente HTTP

Ahora que ya conocemos un poco sobre el funcionamiento del protocolo HTTP y de como funciona este en el servidor y en el cliente, hablemos ahora, del servidor y del cliente.

Servidor: En una red, es un programa que proporciona algún tipo de servicios a varios usuarios.

Un ***servidor*** de HTTP es un programa encargado de responder a requisiciones que se efectúen en el puerto 80, según lo define el protocolo HTTP. Dado el URL de una requisición, el servidor debe encontrar el recurso, archivo o programa que corresponde a dicho URL y, si es necesario, ejecuta los programas ***CGI*** (Common Gateway Interface) y ***SSI*** (Server Side Includes) asociados a dicha página. Un programa CGI se ejecuta localmente en el servidor y normalmente produce información en HTML o algún otro de los formatos del Web; dicha información le es servida al cliente como respuesta a su requisición. Asimismo, dentro de una página de HTML pueden existir comandos que al ser ejecutados produzcan información que sea incluida en el texto de HTML en que aparecen. El servidor también se encarga de labores de ***autentificación*** y seguridad en los casos en que así sea requerido.

CGI: Son programas que se ejecutan normal mente en respues ta a una requisi ción del usuario del Web.

SSI: Son programas que son ejecutados durante el acceso a una página de Web.

Ahora bien, un ***cliente*** de HTTP es normalmente un navegador de Web tal como Mosaic o Netscape y se encarga de desplegar la información en los formatos de uso más común y de llamar aplicaciones auxiliares para el despliegue de información para los formatos que no son tan convencionales, como lo es Post Script. Los clientes más avanzados ***"cachean"*** información localmente para evitar accesos repetidos y además permiten el uso de ***proxies***.

Cliente: Programa que solicita algún tipo de servicio a un servidor.

Proxies: Sistemas que permiten almacenar información para reducir el tráfico externo de páginas en el Web.

Selección de un servidor HTTP

WWW es la parte del Internet que más usuarios atrae y como resultado el número de compañías interesadas en proveer servidores para HTTP ha crecido. Sin embargo, cada proveedor incluye características únicas a su servidor para tratar de hacerlo más atractivo en un mercado que empieza a ser ya bastante competido (algunos servidores serán más simples que otros, otros más rápidos, otros gratuitos, otros más fáciles de instalar, otros generarán mejores estadísticas de uso, etc.).

Criptografía de llave pública: Mé todo de encrip tamiento.

DES (Data Encryption Standard): Técnica de encriptamiento.

La mayor parte de los sitios de WWW utilizan servidores de dominio público (gratuitos). Según estimaciones, los servidores del NCSA y CERN (Centro Europeo de Física de Partículas) se disputan más de 70% del mercado. Ambos son robustos y proporcionan características que satisfacen las necesidades de la mayor parte de los sitios no comerciales: son rápidos y ofrecen diversos niveles de acceso a las páginas (passwords). Netscape Communications es la opción más popular, lo que le da cierta ventaja ante otras empresas, además podría implementar funciones aún no estandarizadas. Su característica más notable es el uso de ***criptografía de llave pública*** y ***DES*** para realizar transacciones en la Red.

La contraparte del software de dominio publico, también existe para los servidores de HTTP. Los servidores HTTP comerciales tienden a hacer más fácil el proceso de instalación y mantenimiento del servidor mediante interfaces gráficas de uso ameno que hacen posible que la instalación no necesite hacerla un experto -quién tendría que modificar archivos de configuración a través de un editor en los servidores gratuitos. Además, por lo regular, ofrecen soporte técnico especializado.

Hablaremos ahora, hablemos brevemente de los servidores mas populares: NCSA, CERN y Netscape.

NCSA HTTPd. Es el servidor gratuito desarrollado por el NCSA de la Universidad de Illinois en Urbana-Champaingn, fue uno de los primeros en su tipo, es pequeño, rápido e incluye muchas funciones para restringir el acceso a la información.

CERN httpd, también llamado W3C HTTPd. Es un servidor gratuito desarrollado originalmente por el Centro del CERN y cuyo desarrollo actual está a cargo del WWW Consortium (W3C). Además de las funciones tradicionales de servidor puede actuar como proxy detrás de una ***firewall*** -con cache para las páginas más frecuentemente visitadas-, y posee niveles básicos de seguridad para la información que éste distribuye.

Firewall. Sistema de seguridad contra ataques externos a la red donde se encuentra dicho sistema. Un mecanismo alterno de seguridad es conocido como "tunneling", el cual opera en el nivel físico de la red.

Servidores de Netscape Communications. Netscape vende dos servidores:

- Netscape Commerce Server. Permite realizar transacciones comerciales en la red -cuando es utilizado junto con el navegador Netscape. Utilizan el ***algoritmo de criptografía*** de llave pública de ***RSA Data Security***, y pretende implementar transacciones seguras -***cifradas***- en la Red. Netscape Commerce Server incluye una interfaz gráfica para su instalación y mantenimiento. Además de correr en Unix se tiene una

versión para Windows NT.

- Netscape Communications Server. No incluye las funciones de criptografía. Por lo que no es conveniente usarlo para realizar transacciones comerciales.

Vale la pena mencionar que Apache es un servidor basado en la versión 1.3 de HTTPd y que rápidamente ha crecido en popularidad. Es gratuito y está siendo desarrollado por el Proyecto Apache, cuyo objetivo es la creación y mantenimiento de un servidor gratuito de WWW compatible con el estándar HTTP.

Una característica interesante de los servidores arriba mencionados es ***NSAPI***, que es una interfaz en ***C*** y ***C++*** para desarrollar aplicaciones que interactúen directamente con el servidor. Además, Netscape está promoviendo el protocolo SSL (Secure Sockets Layer) que pretende dar seguridad a nivel "***sockets***" en TCP/IP. Actualmente SSL es un ***Internet Draft*** en vías de estandarización. SSL proveerá "sockets" ***tipo Berkeley*** que funcionen igual a los actuales, pero con tres propiedades adicionales: 1) asegurar la privacidad de cada canal de comunicación a través de ***criptografía simétrica***, 2) autentificar el canal a través de criptografía de llave pública, 3) verificar la integridad de los mensajes que viajan a través del canal, encima de lo que ya provee TCP. Por lo anterior, SSL puede gestar un Internet confiable. El Netscape Educacional Server Program permite de manera gratuita el uso de sus servidores a universidades, estudiantes, profesores y organizaciones sin fines de lucro.

Las características técnicas de que hablamos arriba son herramientas importantes para tomar una decisión sobre que servidor HTTP elegir, sin embargo, la respuesta, depende de muchos otros factores como son dinero

Algoritmo de criptografía: Conversión de datos a un código especial, con la finalidad de mantener indescifrable la información en la transmisión en una red pública.

RSA Data Security: Método de encriptamiento de alta seguridad.

Cifrar. Al igual que encrip tar, es la conver sión de datos a un código especial, con la finalidad de mantener indesci frable la informa ción en la transmi sión en una red pública.

Internet Draft: Son documentos de la Internet Engineering Task Force (IETF), válidos hasta por 6 meses.

disponible para comprarlo, sistema operativo, tipo de transacciones que se realizarán con el servidor -transferencia de información crítica, por ejemplo-, cantidad de información a transferir, etc.

Por cuestiones económicas se piensa en elegir un servidor de dominio publico: NCSA HTTPd versión 1.4.2, cubre con las características de sencillez y rapidez y proporciona los niveles de seguridad que requerimos en el sistema para el PAPIIT.

Sockets: Función BSD de UNIX que permite a una aplicación acceder a un protocolo de comunicación, crea un canal de comunicación y declara un lugar destino.

Criptografía simétrica: Método de encriptamiento.

Instalación del servidor NCSA HTTPd

Como hemos decidió usa NCSA HTTPd, será necesario dar de alta el servidor paso a paso. A continuación explicaremos el procedimiento:

Para instalar el servidor HTTPd, hay que obtener del programa correspondiente al mismo servidor, a través de la página de HTTPd, el cual hay que ***descomprimir*** (gunzip) y ***separar*** (tar -xvf). El resultado de este proceso es un directorio llamado httpd_1.2.4-solaris, el cual contiene varios subdirectorios y archivos. El archivo más importante es el Makefile, pues al correrlo la primera ocasión desplegará la lista de sistemas operativos soportados. Si lo ejecutamos una segunda vez el Makefile y como argumento se le proporciona el nombre del sistema operativo ("make nombre_del_S.O."), se ***compilarán*** todos los archivos necesarios para que funcione el servidor (NCSA distribuye también versiones ya precompiladas para varios sistemas operativos).

Descomprimir: Expansión de archivos a su tamaño original.

Separar: Reubicación de archivos en sus respectivos directorios según su arborescencia.

Compilar: Software que traduce lenguajes de alto nivel a lenguaje máquina.

Una vez realizados estos pasos es necesario modificar los archivos de configuración, los cuales siguen las siguientes reglas: 1) es indistinto el uso de mayúsculas o minúsculas (excepto para los nombres de directorios y archivos), 2) los comentarios empiezan con #, 3) los espacios extras se ignoran y 4) cada línea debe contener a los más una directiva. Las modifica-

ciones deben abarcar los siguientes puntos:

- Configuración del servidor. El archivo httpd.conf-dist se copia como httpd.conf, este último será el que utilice el servidor.
- ServerType. Si bien el servidor puede correr a través de ***inetd***, es mejor utilizar standalone, pues en el primero ejecutará una copia del servidor por cada requisición y en la segunda se ejecutará una sola vez y permanecerá siempre en memoria. Para la aplicación que vamos a implantar correremos el servidor HTTP como standalone.
- Port. El estándar es utilizar el 80. HTTPd puede ser ejecutado por cualquier usuario si se utiliza un puerto arriba del 1024.

Inetd es un proceso en Unix (daemon o demonio) que administra los servicios que proporciona una máquina donde residen los servidores (ftp, mail, etc.).

User. Para maximizar la seguridad es importante crear un usuario en la máquina donde se instalará el servidor, que sea dueño de los archivos de configuración, y que por tanto tenga una entrada en el archivo /etc/passwd. Se piensa en crear una cuenta llamada http, que sea dueña de los procesos y archivos necesarios para que funcione correctamente el servidor.

```
...
noaccess:x:60002:60002:uid no access:/:
sybase:x:20:60001:Clave de Sybase:/opt/sybase:/bin/csh
ricardo:x:21:15:Ricardo Teapila Cuateco:/home/ricardo:/bin/csh
http:x:22:15:Servidor http:/home/http:/bin/csh
juan:x:23:15:Juan Bautista Martinez:/home/juan:/bin/csh
claudia:x:26:15:Claudia Diaz Manzanares:/home/claudia:/bin/csh
...
```

- Group. Al igual que *user* debe tener su correspondiente registro en el archivo

```
...
noaccess::60002:
sistemas::15:
depest::16:
papiit::17:
...
```

ServerAdmin. Así mismo debe existir un nombre de cuenta asignado a quien será el administrador del sitio de WWW.

```
admhttp@tlaloc.dgapa.unam.mx
```

- ServerRoot. Se aconseja utilizar el directorio que HTTPd recomienda para instalar el servidor (usr/local/etc/httpd), nosotros vamos a instalar el servidor en el directorio /home/http/http_1.2.4.
- Configuración de recursos. El archivo srm.conf-dist hay que copiarlo como srm.conf , en él se configurarán los recursos del servidor. Tanto para buscar páginas de WWW como programas CGI el servidor usa un default, sin embargo pueden ser modificados si es necesario. Para la aplicación que estamos desarrollando usaremos para las páginas de WWW el directorio /www, mientras que para los programas CGI definiremos como directorio de trabajo a /home/http/http_1.2.4/cgi-bin.
- Configuración de Acceso. El archivo access.conf-dist debe copiarse como access.conf, el cual será utilizado para restringir el acceso a las páginas del servidor.

Ahora bien, con la clave de ***superusuario*** (root) hay que reubicar los archivos del servidor en los directorios de trabajo definitivos. Por tanto se ejecutará un ***script*** con los siguientes comandos.

Superusuario: Clave de acceso en el sistema operativo UNIX, que posee todos los privilegios para manipular los recursos del sistema.

Script: Programa que contiene una lista de comando y al ser invocado los ejecuta uno a uno.

```
echo "Creando directorio donde residirá el servidor"
mkdir /home/http/httpd_1.2.4
echo "ubicandonos en el directorio donde residirá el servidor"
cd /home/http/ httpd_1.2.4
echo "Creando los directorios que el servidor requiere para su adecuado funcionamiento"
mkdir conf
mkdir cgi-bin
mkdir cgi-src
mkdir logs
mkdir support
mkdir scr
mkdir icons
echo "Copiando la información generada al descomprimir y desempacar el programa"
echo "correspondiente al servidor
cp /home/http/httpd_1.2.4-solaris/conf/* conf
cp /home/http/httpd_1.2.4-solaris/cgi-bin/* cgi-bin
cp /home/http/httpd_1.2.4-solaris/cgi-src/* cgi-src
cp /home/http/httpd_1.2.4-solaris/logs/* logs
cp /home/http/httpd_1.2.4-solaris/support/* support
cp /home/http/httpd_1.2.4-solaris/src/* src
cp /home/http/httpd_1.2.4-solaris/icons/* icons
cp /home/http/httpd_1.2.4-solaris/httpd
echo "Asignando dueño y grupo de HTTP a los directorios"
chown http *
chgrp sistemas *
echo "Creando, asignando dueño y grupo al directorio donde residiran los documentos"
echo "HTML"
mkdir /www
chmod  go+rx /www/papiit
chown papiit /www/papiit
chgrp sistemas papiit
```

El servidor podrá correrse al terminar todo el proceso antes descrito únicamente con la instrucción:

```
httpd
```

estando en el directorio /home/http/httpd_1.2.4. Después, hay que verificar el funcionamiento desde cualquier visualizador, así mismo hay que revisar que las ***ligas*** a otros URL's y directorios funcionen.


Liga: Enlace a otro documento en HTML.

Para hacer que el servidor se ejecute cada vez que la máquina arranque, se deben incluir las siguientes líneas en el archivo /etc/rc.local

```
#Httpserver
/home /http/httpd_1.2.4/httpd
```

En el servidor levantaremos el servidor al ejecutar el script siguiente:

```
./httpd -d /home/http/httpd_1.4.2 -f /home/http/httpd_1.4.2/conf/httpd.conf
```

Existe un archivo que almacena el ***número de proceso*** asociado al servidor cuando se ejecutó su correspondiente programa. Este archivo es el httpd.pid ubicado en el directorio /home/http/http_1.2.4/logs. El contenido del archivo httpd.pid aparece con información como la siguiente:


Número de Proceso: Número de identificación con que UNIX reco noce los procesos que están siendo ejecutados en ese momento.

```
231
```

Por lo tanto, para dar de baja el servidor HTTPd se podría ejecutar la siguiente instrucción:

```
Kill -9 < cat /home/http/httpd_1.2.4/logs/httpd.pid
```

Seguridad

Seguridad en World Wide Web (WWW)

La seguridad de un sistema de cómputo conectado a una red está determinada por muchos más factores que en una máquina aislada. Pero, el tema de seguridad en WWW es inagotable, pues son muchas las formas en las que podríamos comprometer la seguridad de una computadora al instalar un servidor de WWW. Empecemos por los usuarios, de los cuales existen tres tipos: los que visitan tranquilamente el servidor, los que intentan ver más de lo permitido, y los que intentan entrar al sistema sin autorización. Sin embargo, para poder considerar a un sistema Unix como seguro, también debe tenerse en cuenta la seguridad interna. Y por lo tanto se debe estar consciente de que instalar un servidor WWW conlleva los siguientes riesgos:

- Que usuarios no autorizados tengan acceso a documentos confidenciales. El principal objetivo de un servidor es hacer información disponible a quien desee leerla o solo para aquellos a quienes el acceso no sea restringido. Así que, si por accidente, permitimos el acceso a archivos confidenciales, el problema se resolvería fácilmente; por ello es conveniente visitar el sitio al que permitimos la entrada y verificar que tipo de información está saliendo al exterior. Actualmente se están desarrollando múltiples mecanismos para evitar que un mecanismo de acceso pueda ser violado por un intruso y con ello lograr acceso ilegítimo a páginas que debiesen ser confidenciales.
- Que la información accesible a través del servidor permitiera a un extraño invadir el sistema y hacer mal uso de la información crítica del mismo.
-

- Que si información confidencial es compartida entre el usuario y el servidor, ésta sea interceptada por terceros que la puedan mal utilizar, pues, a menos que se tengan los cuidados necesarios, la información que comparten el servidor y el navegador no está cifrada y, por lo tanto, es legible para cualquiera que la intercepte.

- Que el servidor permita a un extraño ejecutar comandos que pongan en riesgo el sistema. Un programa CGI que acepta parámetros de un usuario es un riesgo potencial para el servidor que lo ejecuta, pues un programa CGI mal escrito o con errores puede permitir ejecutar otros programas, que puedan permitir el acceso no autorizado o dañar el sistema (borrar archivos, por ejemplo). De aquí que hay que verificar que los programas CGI que deseemos instalar en nuestro servidor sean seguros.

Debido a que WWW es un mercado de gran potencial dentro de Internet, hay mucho interés en desarrollar un medio ambiente en que se puedan realizar transacciones seguras, permitiendo que la información de un servidor sea accesible sólo por los que deben y que la información intercambiada entre el usuario y el servidor se mantenga confidencial. Al parecer una de las tantas respuestas se encuentra en criptografía de llave pública, que permite cifrar (encriptar) información y además ***autentificar*** a un usuario.

Autentificar: Identificar a un usuario en una red.

Seguridad en UNIX para WWW

Ahora bien, el servidor HTTP trabaja sobre el sistema operativo UNIX, por tanto, hablemos un poco de los aspectos de seguridad que un sistema UNIX debe proporcionar directamente al trabajar con la red y con servidores de Web.

Puertos de TCP/IP

En el conjunto de protocolos ***TCP/IP***, que son casi universalmente usados en Internet, se define el concepto de puertos para la diferenciación de servicios. Este concepto no se refiere a puertos físicos (conexiones en una máquina), sino a puertos lógicos, desde o hacia los cuales se establece una conexión. Así, toda conexión TCP/IP está únicamente identificada por cuatro números

- Dirección de origen (32 bits)
- Puerto de origen (16 bits)
- Dirección de destino (32bits)
- Puerto de destino (16 bits)

En Unix, se cuenta con 65,536 posibles puertos (16 bits), de los cuales los puertos numerados del 0 al 1023 son conocidos como "puertos confiables", debido a que solamente los programas ejecutados por el superusuario pueden establecer conexiones a través de ellos. Esto pretende que un usuario normal no pueda obtener información privilegiada. Sin embargo, es importante recordar que esto es solamente una convención de Unix y no un estándar de TCP/IP, por lo que en una máquina no Unix, como una PC con una tarjeta de red, podría generar conexiones falsas desde puertos confiables. Los servicios de TCP/IP tienen asignados puertos estándares, así, por ejemplo, el servicio de WWW tiene asignado el puerto 80. Si existen varias conexiones al Web, todas ellas son al puerto 80 del servidor, sin embargo, dichas conexiones se originan en puertos diferentes en los clientes, por lo que cada conexión pueden ser manejada de manera separada.

La asociación entre los puertos y servicios se configuran, en Unix, en el archivo /etc/services, del cual se muestra un fragmento:

```
...
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
http            80/tcp          httpd
...
```

Es importante aclarar que cuando se corre el servidor de HTTP como standalone, esta línea no aparece en el archivo /etc/services, pues constantemente esta corriendo el demonio del servidor de Web. En cambio, si quisiéramos que el demonio del servidor HTTP corriera únicamente cuando fuera invocado y que UNIX controlara este servicio, tendríamos que tener esta entrada como se muestra en el ejemplo. Nuestro servidor de HTTP, dará servicio como una máquina standalone.

Demonio Inetd

Demonios: Programas de UNIX que se encuentra corriendo y en espera de ejecutar una función especifica cuando son invocados.

Se cuentan con programas (conocidos en Unix como ***daemons*** o ***demonios***) para proporcionar cada uno de los servicios, como telnet, ftp, finger, y por supuesto HTTP. Con el paso del tiempo, el número de servicios de red ha crecido considerablemente. Por lo que, el tener corriendo siempre los daemons para cada uno de los servicios, en espera de conexiones a dicho servicio, se consideró impráctico y se decidió utilizar un sistema "intermediario", que es utilizado ahora de manera universal en Unix, y se basa en un daemon llamado inetd (Internet services daemon) que es quién está ejecutándose de manera constante, en espera de peticiones de servicios. Cuando se recibe una petición, inetd ejecuta el daemon correspondiente al servicio solicitado, lo "conecta" al puerto apropiado y reanuda su tarea de escuchar.

Inetd funciona con base en una archivo de configuración, almacenado en /etc/inetd.conf en la gran mayoría de los sistemas Unix. Este archivo inetd.conf tiene el siguiente formato:

```
servicios socket protocolo wait/nowait usuario programa args
```

donde servicios es el nombre del servicios establecido en esa línea; socket es el tipo de socket que utiliza el servicio, normalmente es ***stream*** o ***datagram***; protocolo es el protocolo que utiliza el servicio, normalmente es tcp o udp; wait/nowait: wait indica que, después de proporcionar un servicio, el servidor en cuestión se queda corriendo por un periodo de tiempo, en espera de otras posibles conexiones, nowait indica que el servidor muere inmediatamente después de proporcionar el servicio; usuario indica el usuario con cuyos privilegios se ejecutará el servidor, y programa args indica el programa a ejecutar para proporcionar este servicio, junto con los argumentos que pudiera necesitar.

Stream o Datagrama: Grupo contiguo de datos, que sirven como unidad de mensaje TCP/IP que contiene las direcciones de ori gen y destino de la Internet y los datos.

Se considera apropiado desactivar, en el archivo inetd.conf, los servicios que no vayan a ser utilizados o que no se desee proporcionar, para evitar tener puntos innecesarios de posible acceso al sistema. Esto se puede hacer comentando las líneas correspondientes, poniéndoles el carácter "gato (#) al principio de la línea, y reinicializando el inetd. A continuación se muestra un fragmento de un archivo /etc/inetd.conf:

```
...
# Syntax for socket-based Internet services:
#  <service_name> <socket_type> <proto> <flags> <user> <server_pathname> <args>
#
# Syntax for TLI-based Internet services:
#
#  <service_name> tli <proto> <flags> <user> <server_pathname> <args>
#
# Ftp and telnet are standard Internet services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/in.ftpd in.ftpd
telnet  stream  tcp     nowait  root    /usr/sbin/in.telnetd    in.telnetd
#
# HTTP server.
#
http    stream tcp      nowait root     /home/http/httpd_1.4.2/httpd
#

...
```

Medidas para incrementar la seguridad en WWW

Veremos a continuación las implicaciones de seguridad en el servicio de WWW, así como posibles técnicas para incrementar la seguridad de los sistemas que lo ofrezcan. Como ya se dijo, WWW es, actualmente, el servicio de Internet más utilizado y la gran mayoría de los documentos en el Web están escritos en HTML (HyperText Markup Language), y son enviados a través de la red utilizando un protocolo conocido como HTTP (HyperText Transfer Protocol). Por lo tanto, todo servidor de WWW debe estar corriendo algún tipo de servidor de HTTP, donde se puede encontrar errores y problemas que permitan acceso no autorizado a los recursos.

Las principales medidas que se deben tomas para incrementar la seguridad de un servidor de WWW se puede resumir en las siguientes:

- Evitar usar un servidor con problemas conocidos. Muchos de los sevidores de HTTP ampliamente utilizados han contenido errores que dan lugar a problemas de seguridad. En la gran mayoría de los casos, estos errores ha sido corregidos en cuanto fueron detectados, y las correcciones incorporadas en versiones posteriores del servidor. Por ello, es recomendable utilizar siempre la última versión del servidor de HTTP que se haya elegido.

Que los permisos en los archivos del servidor sean correctos. Tanto los archivos del servidor (el ejecutable, archivos de configuración, etc.), como los documentos almacenados en él deben tener permisos de acceso cuidadosamente establecidos, para impedir que alguna persona no autorizada, ya sea accidental o intencionalmente, haga modificaciones que pudieran causar problemas. En términos generales es conveniente:

- Crear un usuario y/o grupo dedicado al servidor de HTTP.
- El programa del servidor de HTTP debe ser ejecutable solamente por el usuario autorizado (puede ser root para poder otorgar el servicios en el puerto 80).
- Los archivos de configuración del servidor deben ser modificables solamente por el usuario autorizado.

- Los documentos deben ser modificables solamente por el usuario y/o grupo autorizado.

Revisar que no sea posible utilizar ligas simbólicas para dar acceso a archivos que están fuera del árbol de documentos del servidor.

- Reducir al mínimo las cuentas de usuarios existentes en la máquina que funciona como servidor.
- Controlar lo que los usuarios ponen en sus páginas personales y concientizarlos de cualquier acto.
- De ser posible, ejecutar el servidor en un ambiente ***chroot*** (con acceso solamente a una parte del sistema de archivos), para que solo tenga acceso a los archivos estrictamente indispensables.

Mail: Servicio de red usado para transferir mensajes entre usuarios de diferentes máquinas.

Chroot: Tipo de acceso al servidor, que limita este a sólo una parte del sistema de archivos.

Monitorear periódicamente las bitácoras del servidor para detectar comportamientos extraños. La bitácora se almacena el archivo /home/http/httpd_1.2.4/logs/logs. Podríamos, entonces, revisar solo los últimos 100 accesos al servidor con una instrucción como la siguiente:

```
tail -100 /home/http/httpd_1.4.2/logs/logs
```

Siendo el contenido del archivo el siguiente:

```
...
132.248.20.83 - - [02/Apr/1996>15>37>53 -0600] "GET /dgapa/dgapa?fu.html HTTP/1.0" 200 1316
pc1.ccp.pty.com - - [02/Apr/1996>16>09>25 -0600] "GET /dgapa/dgapa?24.html HTTP/1.0" 200
4337
lemarec.anx.rmc.ca - - [02/Apr/1996>16>30>38 -0600] "GET /dgapa/conv5?04.html HTTP/1.0" 200
5493
lemarec.anx.rmc.ca - - [02/Apr/1996>16>30>46 -0600] "GET /imagenes/bk?dgapa.gif HTTP/1.0"
200 5838
lemarec.anx.rmc.ca - - [02/Apr/1996>16>31>02 -0600] "GET /imagenes/hr?dgapa.gif HTTP/1.0"
200 2541
132.254.54.12 - - [02/Apr/1996>17>09>09 -0600] "GET /dgapa/supc?a01.html HTTP/1.0" 200 3163
132.254.54.12 - - [02/Apr/1996>17>09>16 -0600] "GET /imagenes/bk?dgapa.gif HTTP/1.0" 200
5838
132.254.54.12 - - [02/Apr/1996>17>09>17 -0600] "GET /imagenes/hr?dgapa.gif HTTP/1.0" 200
2541
132.254.54.12 - - [02/Apr/1996>17>09>19 -0600] "GET /imagenes/estadist.gif HTTP/1.0" 200 3856
...
```

- No confiar en restricciones de acceso por dirección IP, o por contraseñas, para proteger documentos confidenciales, pues estas mediadas pueden ser violadas por un experto.

Desactivar la ejecución de programas CGI (Common Gateway Interface) cuando estos no sean necesarios y cuidadosamente controlados. Los programas CGI´s se pueden activar o desactivar dando o eliminando todos los permisos de escritura, lectura y ejecución sobre los directorios donde se encuentran dichos programas, esto es,

```
eliminar acceso
chmod 700 /home/http/httpd_1.4.2/cgi-bin/renovación
proporcionar acceso
chmod 744 /home/http/httpd_1.4.2/cgi-bin/renovación
```

- Revisar minuciosamente los programas CGI que se utilicen, para que no se ejecuten con ninguna clase de privilegios, no modifiquen el servidor y no utilicen ninguna clase de información proporcionada por el usuario sin antes revisarla muy detalladamente.

Finalmente, también es importante tener cuidado del lado del cliente, pues, si el visualizador que se está utilizando está configurado para ejecutar "ciegamente" cualquier documento de tipo Aplicación que se encuentre, es posible ejecutar comandos arbitrarios en la máquina en la que se está ejecutando dicho visualizador.

En general, estas son las medidas que se pueden tomar para proporcionar un rango mayor de seguridad en los servidores HTTP

Proxies y Caching

Con el incremento en la popularidad del Web, Internet se ha visto sobrecargada. Por ejemplo, una página con un número moderado de imágenes puede ocupar un enlace de 64 kbps por unos 20 segundos sin permitir ningún otro tráfico durante este intervalo. Por ello se ha dado varias propuestas para reducir el tráfico en la red. Una de las más comunes es el uso de ***caché*** en un ***proxy*** para almacenamiento local de páginas del Web. Este proxy reside en una computadora que comúnmente está localizada en la frontera de la red local y la red global. Con esto, el objetivo es, por un lado, aislar computadora locales de la red global y, por otro almacenar información para reducir el tráfico externo de páginas Web. Es posible instalar una PC como proxy, y con 2 gigabytes de disco puede almacenar aproximadamente de 10,000 a 20,000 páginas. Cuando se acaba el espacio, las páginas de menor uso son las primeras en ser removidas, en tanto que las de uso común se mantienen por tiempos más largos, reduciendo así el tráfico externo. Las computadoras proxy son particularmente útiles en organizaciones grandes, tales como universidades o ***proveedores de acceso a Internet*** (***ISP***), que tienen un sin número de computadoras conectadas al gateway principal. La nueva versión de HTTP soporta el modificador de comando "if-modified-since", que verificar la validez de la página en caché local. Si la página ha sido modificada desde el día que fue "cacheada", se baja una nueva copia de la página, si ésta no se ha cambiado el proxy sirve al cliente la página localmente.

Proxies: Sistemas que permiten almacenar información para reducir el tráfico externo de páginas en el Web.

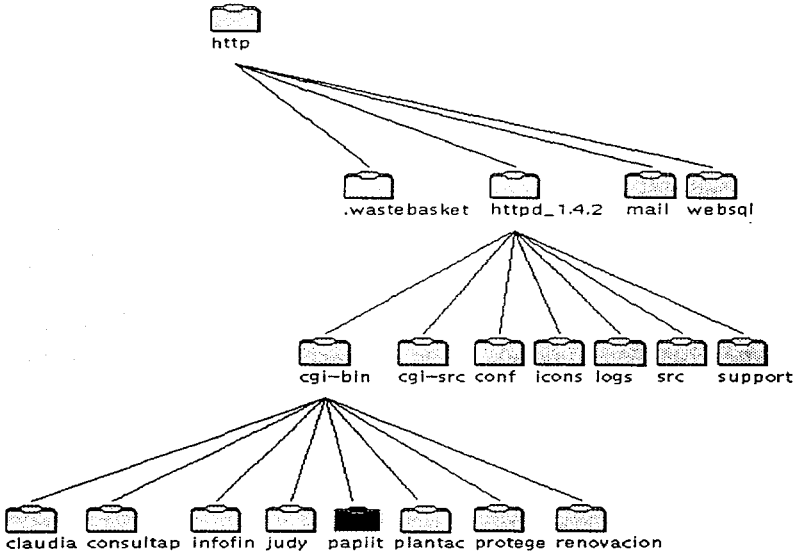
Proveedores de acceso a Internet (ISP): Compañías privadas que ofrecen el servicio de conexión a Internet a cambio de un pago por el servicio.

Asimismo, el proxy actúa como "representante" del cliente. Cuando el cliente requiere un documento, ya sea por ftp, http o gopher, le pasa la solicitud al proxy el cual a su vez verifica si la tiene en el caché, si éste es el caso sirve el documento al cliente y si no, lo solicita a la dirección indicada y lo recibe en modo cliente y una vez que el documento se encuen

tra en el caché local es servido al requisidor original.

Configuración del servidor HTTP

Como ya dijimos, la versión del servidor HTTP que utilizaremos para desarrollar nuestra aplicación es la 1.4 y los archivos de configuración residen como se muestra en la siguiente estructura arborescente



- **Directorio /home/http/httpd_1.4.2**

Residen dos archivos ocultos de acceso, específicos para cada aplicación que se utilice en el servidor HTTP. Uno de ellos es el archivo que contendrá un grupo de usuarios, nosotros le llamaremos .grupo.papiit, mientras que el otro contiene el login y el password encriptado de los usuarios autorizados en el grupo (.grupo.papiit) y se le determinará con el siguiente nombre .passwd.solicitud. El contenido de ambos será similar a lo siguiente:

Archivo .grupo.solicitud

```
...
usuarios-papiit: IN200297
usuarios-papiit: IN200497
usuarios-papiit: IN100697
usuarios-papiit: IN100497
usuarios-papiit: IN200697
usuarios-papiit: IN300297
usuarios-papiit: IN200897
...
```

Archivo .passwd.papiit

```
...
IN200297:dWk8MYIlXtQyQ
IN200497:LFSnSsxwsZ3aw
IN100697:DhdP.3hX1aJwU
IN100497:y/h8upX4odnAA
IN200697:TvEhKwdIf76Xo
IN300297:IrpONJB7H1aPQ
IN200897:z69LVOOUhjG5Q
...
```

El servidor HTTP posee una utilería llamada htpasswd que crea estos archivos y que además actualiza las entradas cuando ingresan nuevos

usuarios. El procedimiento para generar dichos archivos lo explicaremos más adelante, cuando tratemos este archivo.

- **Directorio conf**

  Directorio que contiene lo archivos de configuración necesarios para que el servidor funcione adecuadamente. De este directorio dependen tres archivos principales:

Archivo access.conf. En él se definen equipos, segmentos o dominios en la red que tendrán acceso al sistema. Esto se especifica en la siguiente parte del archivo de la siguiente manera:

```
...
#<Directory /www/directorio_de_acceso>
<Limit GET>
order deny-allow
deny from all
allow from <[dominio,segmento,usuario ] >
....
```

ejemplo:

```
# access.conf: Global access configuration
# Online docs at http://hoohoo.ncsa.uiuc.edu/
# I suggest you consult them; this is important and confusing stuff.

# /usr/local/etc/httpd/ should be changed to whatever you set ServerRoot to.
<Directory /usr/local/etc/httpd/cgi-bin>
Options Indexes FollowSymLinks
```

```
</Directory>

# This should be changed to whatever you set DocumentRoot to.

<Directory /usr/local/etc/httpd/htdocs>

# This may also be "None", "All", or any combination of "Indexes",
# "Includes", or "FollowSymLinks"

Options Indexes FollowSymLinks

# This controls which options the .htaccess files in directories can
# override. Can also be "None", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"

AllowOverride All

# Controls who can get stuff from this server.

<Limit GET>
order allow,deny
allow from all
</Limit>
</Directory>

# You may place any other directories you wish to have access
# information for after this one.

#Restricciones para la solicitud
<Directory /home/http/httpd_1.4.2/cgi-bin/solicitud>
<Limit GET>
order deny,al'ow
deny from all
allow from 13.2.248.37.
allow from 132.248.11.22
</Limit>
</Directory>
```

```
#Restricciones para la renovación
<Directory /home/http/httpd_1.4.2/cgi-bin/renovacion>
<Limit GET>
order deny,allow
deny from all
allow from 132.248.37.
</Limit>
</Directory>
```

Archivo srm.conf. Contiene el nivel de búsqueda de documentos HTML por default. Con nivel nos referimos a partir de que directorio dentro de la arborescencia se podrán realizar dichas búsquedas. En lugar de un directorio definiremos un archivo html:

```
# With this document, you define the name space that users see of your http
# server.

# See the tutorials at http://hoohoo.ncsa.uiuc.edu/docs/tutorials/ for
# more information.

# NCSA httpd (httpd@ncsa.uiuc.edu)


# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.

DocumentRoot /www

# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is recieved.
```

```
UserDir public_html

# DirectoryIndex: Name of the file to use as a pre-written HTML
# directory index

DirectoryIndex dgapa/dgapa.html

# FancyIndexing is whether you want fancy directory indexing or standard

FancyIndexing on

# AddIcon tells the server which icon to show for different files or filename
# extensions

AddIconByType (TXT,/icons/text.xbm) text/*
AddIconByType (IMG,/icons/image.xbm) image/*
AddIconByType (SND,/icons/sound.xbm) audio/*
AddIcon /icons/movie.xbm .mpg .qt
AddIcon /icons/binary.xbm .bin

AddIcon /icons/back.xbm ..
AddIcon /icons/menu.xbm ^^DIRECTORY^^
AddIcon /icons/blank.xbm ^^BLANKICON^^

# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.

DefaultIcon /icons/unknown.xbm

# AddDescription allows you to place a short description after a file in
# server-generated indexes.
# Format: AddDescription "description" filename

# ReadmeName is the name of the README file the server will look for by
# default. Format: ReadmeName name
#
# The server will first look for name.html, include it if found, and it will
```

```
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.

ReadmeName README
HeaderName HEADER

# IndexIgnore is a set of filenames which directory indexing should ignore
# Format: IndexIgnore name1 name2...

IndexIgnore */.??* *~ *# */HEADER* */README*

# AccessFileName: The name of the file to look for in each directory
# for access control information.

AccessFileName .htaccess

# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.

DefaultType text/plain

# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
# Format: AddType type/subtype ext1

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.

#AddEncoding x-compress Z
#AddEncoding x-gzip gz

# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect fakename url
```

```
# Aliases: Add here as many aliases as you need, up to 20. The format is
# Alias fakename realname

Alias /icons/ /home/http/httpd_1.4.2/icons/

# ScriptAlias: This controls which directories contain server scripts.
# Format: ScriptAlias fakename realname

ScriptAlias /cgi-bin/ /home/http/httpd_1.4.2/cgi-bin/

# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.

#AddType text/x-server-parsed-html .shtml
#AddType application/x-httpd-cgi .cgi

# If you want to have files/scripts sent instead of the built-in version
# in case of errors, uncomment the following lines and set them as you
# will.  Note: scripts must be able to be run as if the were called
# directly (in ScriptAlias directory, for instance)

# 302 - REDIRECT
# 400 - BAD_REQUEST
# 401 - AUTH_REQUIRED
# 403 - FORBIDDEN
# 404 - NOT_FOUND
# 500 - SERVER_ERROR
# 501 - NOT_IMPLEMENTED

#ErrorDocument 302 /cgi-bin/redirect.cgi
#ErrorDocument 500 /errors/server.html
#ErrorDocument 403 /errors/forbidden.html
```

El directorio que se definió es /www

de él dependerán otros directorios a los que también se podrá accesar, y corresponderán a cada una de las aplicaciones que se desarrolle, así por ejemplo tendremos que el directorio que contendrá información HTML del PAPIIT será /www/papiit.

- **Archivo httpd.conf.** Es el archivo principal de configuración del servidor.

```
# This is the main server configuration file. It is best to
# leave the directives in this file in the order they are in, or
# things may not go the way you'd like. See URL http://hoohoo.ncsa.uiuc.edu/
# for instructions.

# Do NOT simply read the instructions in here without understanding
# what they do, if you are unsure consult the online docs. You have been
# warned.

# NCSA httpd (comments, questions to httpd@ncsa.uiuc.edu)

# ServerType is either inetd, or standalone.

ServerType standalone

# If you are running from inetd, go to "ServerAdmin".

# Port: The port the standalone listens to. For ports < 1023, you will
# need httpd to be run as root initially.

Port 80

# StartServers: The number of servers to launch at startup.  Must be
# compiled without the NO_PASS compile option
```

```
StartServers 5

# MaxServers: The number of servers to launch until mimic'ing the 1.3
# scheme (new server for each connection).  These servers will stay around
# until the server is restarted.  They will be reused as needed, however.
# See the documentation on hoohoo.ncsa.uiuc.edu for more information.

MaxServers 20

# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.

# User/Group: The name (or #number) of the user/group to run httpd as.

User http
Group staff

# ServerAdmin: Your address, where problems with the server should be
# e-mailed.

ServerAdmin ricardo@tlaloc.dgapa.unam.mx

# ServerRoot: The directory the server's config, error, and log files
# are kept in

ServerRoot /home/http/httpd_1.4.2

# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog logs/error_log

# TransferLog: The location of the transfer log file. If this does not
# start with /, ServerRoot is prepended to it.

TransferLog logs/access_log
```

```
ServerRoot /home/http/httpd_1 4 2

# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog logs/error_log

# TransferLog: The location of the transfer log file. If this does not
# start with /, ServerRoot is prepended to it.

TransferLog logs/access_log

# AgentLog: The location of the agent log file.  If this does not start
# with /, ServerRoot is prepended to it.

AgentLog logs/agent_log

# RefererLog: The location of the referer log file.  If this does not
# start with /, ServerRoot is prepended to it.

RefererLog logs/referer_log

# RefererIgnore: If you don't want to keep track of links from certain
# servers (like your own), place it here.  If you want to log them all,
# keep this line commented.

#RefererIgnore servername

# PidFile: The file the server should log its pid to
PidFile logs/httpd.pid

# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e. use
# "www" instead of the host's real name).
#
```

```
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.

#ServerName new.host.name
ServerName tlaloc.dgapa.unam.mx
```

- **Directorio cgi-bin**

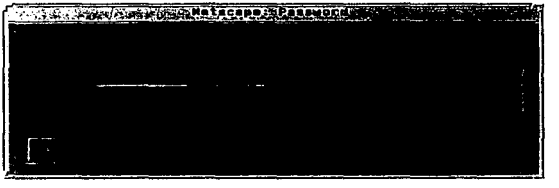
Directorio que contendrá únicamente archivos ejecutables que serán utilizados en nuestro sistema para decodificar las formas HTML que utilicen el método POST (CGI's de usuario). Se recomienda crear un directorio sobre cgi-bin para cada programa que se realice, por ejemplo, existirán algunos directorios como papiit, pruebas, plantac, etc. sobre cgi-bin. Para nuestro sistema definiremos crearemos el directorio /home/http/httpd_1.4.2/cgi-bin/papiit. Es importante hacer notar que cada directorio sobre cgi-bin tiene un archivo llamado .htaccess, el cual define que usuarios van a tener acceso a los programas ubicados en el directorio, su contenido lo veremos a continuación:

```
...
AuthUserFile <directorio_de_trabajo>/<archivo_de_passwords>
AuthGroupFile <directorio_de_trabajo>/<archivo_de_grupo_de_usuarios>
AuthName<encabezado_para_caja_de_dialogo_donde_se_pide_el_password>
AuthType Basic

<Limit GET>
requiere group <grupo_especificado_en_el_archivo_de_grupo_de_usuarios>
</Limit>
...
```

ejemplo,

```
AuthUserFile /home/http/httpd_1.4.2/.passwd.papiit
AuthGroupFile /home/http/httpd_1.4.2/.grupo.papiit
AuthName NUMERO DE PROYECTO PAPIIT
AuthType Basic

<Limit GET>
require group usuarios-papiit
</Limit>
```

Lo que se vera reflejado en la caja de diálogo de la siguiente manera:



- **Directorio cgi-src**

  En este directorio podemos localizar todos los archivos ejecutables que decodifican las formas (CGI's del servidor) que utiliza exclusivamente el servidor HTTP.

- **Directorio logs**

  En él se ubica la auditoria al servidor, por ello se encuentra ahí uno de los archivos más útiles:

  - **Archivo access_log**. En dicho archivo podemos consultar de una manera detallada el acceso al servidor. El contenido se puede ver con la siguiente instrucción:

```
tail -100 access_log
```

del mismo posee la siguiente forma:

```
...
pc1.ccp.pty.com - - [02/Apr/1996>16>09>25 -0600] "GET /dgapa/dgapa?24.html
HTTP/1.0" 200 4337
lemarec.anx.rmc.ca - - [02/Apr/1996>16>30>38 -0600] "GET /dgapa/conv5?04.html
HTTP/1.0" 200 5493
lemarec.anx.rmc.ca - - [02/Apr/1996>16>30>46 -0600] "GET /imagenes/bk?dgapa.gif
HTTP/1.0" 200 5838
lemarec.anx.rmc.ca - - [02/Apr/1996>16>31>02 -0600] "GET /imagenes/hr?dgapa.gif
HTTP/1.0" 200 2541
132.254.54.12 - - [02/Apr/1996>17>09>09 -0600] "GET /dgapa/supe?a01.html HTTP/1.0"
200 3163
132.254.54.12 - - [02/Apr/1996>17>09>16 -0600] "GET /imagenes/bk?dgapa.gif
HTTP/1.0" 200 5838
132.254.54.12 - - [02/Apr/1996>17>09>17 -0600] "GET /imagenes/hr?dgapa.gif
HTTP/1.0" 200 2541
132.254.54.12 - - [02/Apr/1996>17>09>19 -0600] "GET /imagenes/estadist.gif HTTP/1.0"
200 3856
...
```

Observemos que nos proporciona información sobre qué maquina realiza la petición, el día y la hora de acceso, el archivo accesado, y su correspondiente numero de identificación.

- **Directorio support**

Contiene utilerías para el servidor HTTP. Una de las más usadas es el programa:

Utilería htpasswd. Es una utilería que permite controlar el acceso al servidor, mediante el uso de passwords. Este programa actúa directamente sobre un archivo oculto llamado, en el caso específico de nuestra aplicación, /home/http/httpd_1.4.2/.passwd.papiit. htpasswd reconoce este archivo mediante una consulta que realiza al archivo .htaccess correspondiente, donde se le indica su uso. Es importante hacer notar, que el archivo oculto (.passwd.papiit) debe encontrarse en el ruta /home/http/httpd_1.4.2

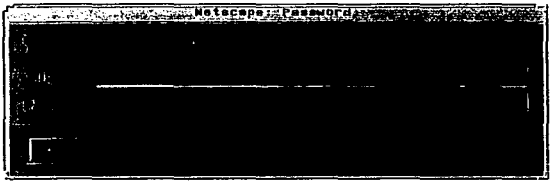
Si el archivo /home/http/httpd_1.4.2/.passwd.papiit no existe, en el comando sería necesario utilizar una opción -c para crearlo, y la sintaxis correspondería a la siguiente (estando en el directorio /home/http/httpd_1.4.2/support):

htpasswd -c ../archivo_oculto login_name

```
htpasswd -c ../.passwd.papiit IN111194
```

Dicha utilería se ejecutará desde un programa que detecte si el archivo ha sido creado o no, además de determinar si la cuenta que ingresa es nueva o ya se dio de alta en el sistema, el programa fuente que automatiza este proceso se puede consultar en el apéndice 3:

El password se introduce en la siguiente pantalla:



El sistema pedirá el login y el password correspondientes a los usuarios y los incluirá automáticamente al archivo. Obsérvese que nuestro sistema empleará como el login de la cuenta el numero de proyecto.

◆ **Directorio src**

Contiene todo el código fuente del servidor HTTP. Dicho código esta escrito en lenguaje C.

Programación bajo WWW

Hoy en día nosotros estamos viviendo en un mundo "multi-plataformas" en el cual cada uno quiere crear aplicaciones que están en plataformas independientes. Pero porque no aprovechar el duro trabajo de otras personas y hacer cosas fáciles para ti mismo usando HTML en el Web. Recordemos que el Web fue originalmente desarrollado para permitir compartir información entre equipos dispersos y promover la divulgación de información; hoy, a unos años de su surgimiento el Web es el mayor sistema de información desarrollado en Internet.

La razón principal para usar HTML como interfaz con el usuario para tus aplicaciones es que por hacer esto estás tomando ventaja del hecho de que existen visualizadores para todas las plataformas que tú probablemente estés interesado en tener tu aplicación "corriendo". En este sentido, cualquier tipo de máquina estará en la posibilidad de ejecutar la aplicación.

Los formularios interactivos son el equivalente a las hojas de papel que nosotros llenamos cada día en toda la vida, como puede ser una solicitud de inscripción a una escuela.

La potencialidad que ofrece el Web es inmensa. Imagínese ¿Tener ***formularios interactivos*** para que los usuarios soliciten información, realizar una inscripción, hacer reservas o pedidos, etc.? Todo esto e incluso más se puede hacer a través de una manera fácil y rápida en el Web.

Usando la tecnología del Web podemos hacer que nuestras aplicaciones sean accedidas desde cualquier parte del mundo a través de Internet.

En este capítulo vamos a describir como construir aplicaciones gráficas usando HTML, Programación de Clientes en Sybase y programación de CGI's. Muchos lenguajes de programación pueden ser usados para escribir los programas, como pueden ser C, C++, PERL y Pascal.

Este documento requiere de un previo conocimiento de creación de documentos HTML. Si tú no tienes este conocimiento por favor repasa el capítulo 6 de este libro.

Elementos necesarios para empezar

- Primero vas a necesitar un Visualizador Web (como puede ser Netscape)
- Vas a necesitar facilidades para generar un código ejecutable: si tu estas escribiendo en un lenguaje compilable (C, C++, PASCAL, etc.), o un parser si tu estas escribiendo en un lenguaje como PERL.
- Tener una cuenta en el servidor de bases de datos Sybase.
- Librerías que permitan la comunicación de un programa cliente con la Base de Datos (DB-Library).
- Una librería para interpretar documentos del Web (en el apéndice de este libro se encuentra el código fuente de dicha librería).
- Un editor de texto que te permita editar los programas, éste editor puede ser *vi* o *pico* para Unix, te recomendamos que uses *pico* por su facilidad.
- Lo último que necesitarás es una computadora corriendo un Servidor Web, y el servidor tiene que permitir correr programas CGI, esto es una cuenta en Unix con permisos. Si tu no estas seguro verifica con tu administrador del sistema, o con una persona que conozca acerca de esto.

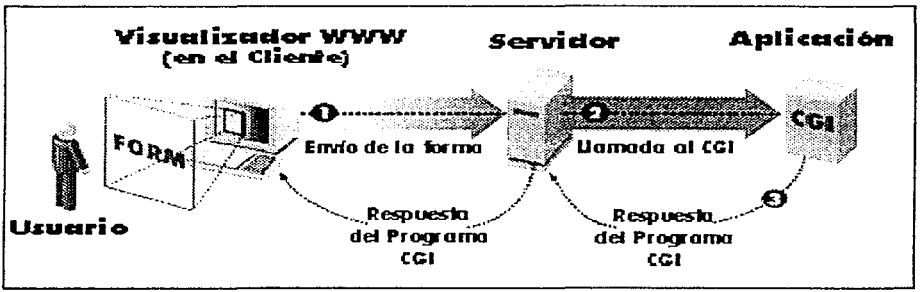
Para poder realizar los ejemplos mostrados en este capítulo es necesario que se creen las tablas con sus datos mostrados al final del capítulo 4 de ete libro.

Cómo ya vimos el World Wide Web (WWW) es el más popular y potente sistema de información en red hasta la fecha. WWW ha tenido un crecimiento exponencial y ha causado una verdadera revolución de información que probablemente continuará en el próximo siglo.

Por otra parte, el mundo de las bases de datos con tan sólo 25 años como un campo básico de investigación no es menos interesante. Es necesario decir, que el encuentro de estos dos mundos trae nuevas oportunidades para la avanzada creación de aplicaciones.

Formas WWW

Un panorama general del desarrollo de aplicaciones en WWW es como se muestra en la siguiente figura:



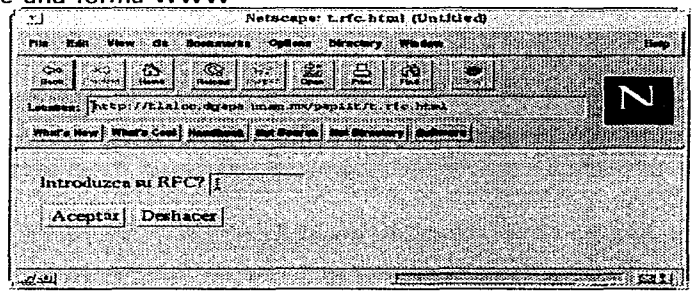
En la figura podemos ver que el usuario, por medio de un visualizador WWW, llena una forma de captura de datos y después la envía al servidor ❶; una vez que llega la forma al servidor, éste manda llamar una aplicación CGI para que atienda la forma ❷; por último la aplicación CGI le dice al servidor de la acción que debe realizarse ❸, éste a su vez es el encargado de dar una respuesta al usuario sobre el resultado de la captura de datos.

A continuación, de acuerdo al panorama general antes mostrado, vamos a mostrar los conceptos de: formas WWW y programación de CGI's. Cuando tu oprimes este botón, dos cosas son enviadas al servidor: el tipo de datos que tu tienes en la forma, y una ACCION, la cual básicamente llama el

nombre del programa quien conoce como procesar los datos de la forma.

La construcción de una forma no es tan difícil como parece al principio. Si tu has visto algo de HTML te será muy sencillo.

ejemplo de una forma WWW



Aquí está el código fuente de la forma WWW:

```
<FORM METHOD = POST ACTION = "http://hoohoo.ncsa.uiuc.edu/htbin-post/post-query" >
Introduzca su RFC: <INPUT TYPE = text NAME = rfc SIZE = 10 VALUE = "" >
<BR >
<INPUT TYPE = reset VALUE = Deshacer >
<INPUT TYPE = submit VALUE = Aceptar >
</FORM >
```

Las etiquetas (tags) en HTML no son sensitivas, esto es, pueden escribirse tanto en letras mayúsculas como en letras minúsculas (INPUT = input). Por convención, recomendamos escribir estas etiquetas siempre con letras mayúsculas.

Las etiquetas FORM

```
<FORM METHOD = POST ACTION = "[nombre del programa CGI]" >
</FORM >
```

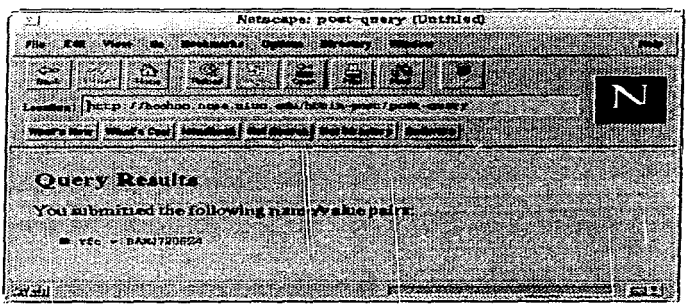
Estás etiquetas inician y finalizan una forma (todos los campos de entrada de la forma son colocados entre estas dos etiquetas).

METHOD especifica la forma técnica que va a usar el servidor Web para pasar los datos al programa que los va a procesar, y ACTION le dice al servidor cual es éste programa.

POST debe especificarse en el METHOD, en otro caso el default es GET. Más adelante veremos más a detalle los métodos POST y GET.

ACTION especifica el nombre y ubicación del programa que va a recuperar los datos enviados desde la forma al servidor. En nuestro ejemplo, el ACTION es un programa llamado "post-query" que está localizado en un servidor de NCSA. Post-query es un programa que proporciona NCSA para probar formas. Este simplemente imprime los nombre y contenidos de todos los campos de entrada de una forma que ha sido enviada.

Supongamos que nosotros dimos el siguiente rfc: BAMJ720624 en el ejemplo anterior, el resultado de enviar la forma usando el programa post-query sería la siguiente:



Una página puede incluir múltiples formas; sin embargo, las formas no deben estar anidadas (no debes crear una forma dentro de otra forma). Tú debes finalizar la actual forma con </FORM> antes de poder iniciar una nueva forma.

Recuperando una línea de texto del usuario

```
Introduzca su RFC: <INPUT TYPE=text NAME=rfc SIZE=10 VALUE="">
```

El anterior código crea la caja de entrada del nombre del usuario en nuestro ejemplo. NAME define el nombre del dato para el campo; esto es como el programa el cual procesa la forma referencia los datos de este campo.

Tú puedes escoger un nombre para tus campos, sin repetirlos en una misma forma. Tu debes ser cuidadoso en usar sólo letras, números y subguiones en los nombres de campos preferiblemente, ya que el HTML estándar no reconoce todos los símbolos. TYPE define la variedad de campos de entrada reconocidos en las especificaciones de HTML (text, radio, checkbox, password, reset o submit). Para mayor referencia ver el capítulo 6 de este libro.

La entrada del tipo *text* crea una caja de entrada de texto que le permite al usuario introducir y/o editar el texto.

El botón submit

```
<INPUT TYPE=submit VALUE=Aceptar>
```

Cada forma debe tener exactamente un campo de tipo submit. Este botón es usado para que el usuario lo oprima una vez que ha terminado de llenar la forma, y debe de ir al final de todos los campos de entrada. Si una forma no tiene un botón submit, está es inservible, pues el usuario no tiene

manera alguna de enviar los datos al servidor.
Tú puedes poner las palabras que quieras, usando la opción VALUE. Si tú omites está opción, el botón va a aparecer etiquetado con "Submit".

Cuando el usuario oprime el botón submit, el visualizador recolecta cada uno de los campos de entrada (texto escrito por el usuario, etc.) y envía a estos al programa especificado en la forma.

El botón reset

<INPUT TYPE = reset VALUE = Deshacer >

Este botón es usado para regresar los valores de la forma a sus valores por default (valores originales). En este ejemplo, el oprimir el botón reset borraría lo escrito y dejaría la caja de texto en blanco.

Con la introducción de las formas en el Web, las bases de datos han tomado una gran importancia en el desarrollo de aplicaciones en WWW, debido a que ahora se requiere de un lugar para almacenar los datos capturados en las formas.

Una pregunta en la que pensaríamos es la siguiente, ¿Cómo es que las aplicaciones del Web pueden introducir, recuperar, cambiar y eliminar datos de una base de datos sin que el usuario se de cuenta? La respuesta a ésta pregunta es la siguiente: existen un conjunto de funciones que permiten interactuar a un programa (como puede ser nuestra forma WWW) directamente con la base de datos, sin la necesidad de utilizar algún otro programa intermediario para la manipulación de datos dentro de la base de datos como son ISQL o WISQL. A este conjunto de funciones se les conoce como DB-Library y forman parte de la Programación de Clientes con Sybase.

 Los ejemplos aquí mostrados son exclusivos de la Programación de Clientes con Sybase en UNIX, esto es, que deben ser compilados con un

compilador desde el sistema operativo UNIX.

Programación de Clientes con Sybase

La Programación de Clientes con Sybase para generar programas ejecutables, llamados genéricamente **aplicaciones clientes** o simplemente **clientes**, que corran bajo World Wide Web (WWW) desplegando datos recuperados de una base de datos en Sybase; para lograr este objetivo, la Programación de Clientes con Sybase utiliza funciones conocidas como DB-Library's de las cuales hablaremos un poco a continuación.

DB-Library

Las DB-Library's son API's (Applications Programming Interface), consisten de varias rutinas utilizadas para manejar la comunicación entre aplicaciones clientes y los servidores de Sybase. Las DB-Library's han sido desarrolladas para los lenguajes C, COBOL, FORTRAN, Ada y Pascal.


Los ejemplos usados en este capítulo están escritos en lenguaje C.

Entre sus múltiples funciones de las DB-Library's se encuentran las siguientes:

- Trasmite sentencias SQL al servidor
- Regresa los datos al programa cliente
- Realiza la conversión de datos

Las DB-Library se componen de:

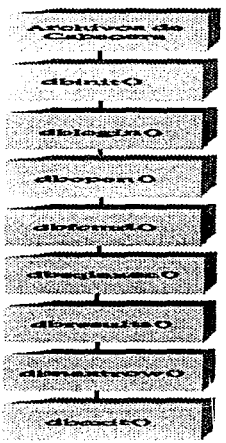
- Un conjunto de funciones que permiten desarrollar aplicaciones clientes.
- Un conjunto de librerías que se encargan de adaptar los protocolos de comunicación, conocidas como Net-Library.

- Un conjunto de librerías del servidor para el caso en que se use un servidor programable (Open-Server) y no un servidor estándar (SQL Server), conocido como Server-Library.

Los beneficios de las DB-Library's son los siguientes:

- Proveen un camino estándar y consistente para los desarrolladores de aplicaciones con los servidores Sybase.
- Aisla a los desarrolladores de los detalles de red y protocolos.
- Provee llamadas de funciones y definiciones estándar (entre las distintas arquitecturas de equipos de cómputo).
- Los clientes son portables entre máquinas de la misma arquitectura.

El esqueleto básico de un programa cliente, que usa las DB-Library, se muestra en la siguiente figura:



Esqueleto básico de un programa cliente

Por convención las funciones que utilizan las DB-Library son de la forma dbxxx.

Archivos de Cabecera. Estos dos archivos, *sybfront.h* y *sybdb.h*, son requeridos en todos los programas que contengan llamadas a las DB-Library's. sybfront.h debe aparecer en primer lugar. Esta define constantes simbólicas, definiciones de tipo (DBCHAR, DBINT, etc.) y los valores de salida STDEXIT y ERREXIT (los cuales son usados en la función *exit()*).
sybdb.h contiene adicionales definiciones de tipos, de las cuales, la más importante es la estructura llamada DBPROCESS. Las DB-Library's se comunican con el servidor a través de una o más estructuras DBPROCESS. Una aplicación puede tener múltiples DBPROCESS's conectados a uno o más servidores.
syberror.h contiene los valores de errores y debe ser incluido si el programa hace referencia a estos errores.

dbinit(). Inicializa las DB-Library **y debe ser la primera rutina de las DB-Library en el programa**.

dblogin(). Crea una estructura llamada LOGINREC que va a ser usada por las DB-Library para conectarse con el servidor. LOGINREC esta compuesta por las macros DBSETLUSER() que especifica el usuario de la base de datos y DBSETLPWD() que especifica el password del usuario.

dbopen(). Abre una conexión entre la aplicación y el servidor SQL. Esta usa la información suministrada por LOGINREC para conectarse al servidor y regresa una estructura llamada DBPROCESS, la cual sirve como conductor de información entre la aplicación y el servidor. Después de llamar ésta rutina, la aplicación esta conectada con el servidor SQL y ahora puede enviar comandos SQL al servidor SQL y procesar los resultados.

dbfcmd(). Llena el ***buffer*** de comandos con comandos SQL, las cuales pueden entonces ser enviados las servidor SQL. Tu puedes incluir múltiples rutinas *dbfcmd()* en el buffer para formar comandos SQL, como puede ser sentencias **select**, **update**, **delete** e incluso **store procedures** (procedimientos almacenados). Cada llamada exitosa de una *dbfcmd()* simplemente agrega el texto suministrado al final de algún texto ya existente en el buffer. Tu debes de tener cuidado en poner espacios entre palabras, principalmente si tu usas múltiples rutinas *dbfcmd()*.

dbsqlexec(). Envía el contenido del buffer de comandos al servidor SQL, el cuál lo verifica y lo ejecuta.

dbresults(). Recupera los resultados del actual comando SQL. Para cada comando en el buffer se necesita llamar a dbresults().

dbbind(). Mapea las columnas (resultantes del comando ejecutado) en variables de programa. Debe de existir un *dbbind()* por cada columna. El mapeo debe corresponder del tipo de dato de la columna con el tipo de variable de programa; sin embargo, dbbind() soporta un rango amplio de conversiones, que permiten recuperar de forma diferente los datos de las columnas.

CONVERSIÓN DE DATOS

Tipo de variable	Tipo de dato en lenguaje C	Tipo de dato en SQL
INTBIND	DBINT	ENTERO
CHARBIND	DBCHAR	CADENA
STRINGBIND		
NTBSTRINGBIND		
FLT8BIND	DBFLT8	REAL
MONEYBIND	DBMONEY	MONEY

CHARBIND rellena los blancos sin el carácter '\0'
STRINGBIND rellena con espacios en blanco.
NTBSTRINGBIND no rellena con espacios.

La función *dbbind()* siempre se llama después de un *dbresulst()* y antes de *dbnextrow()*. La sintaxis de *dbbind()* es la siguiente:

dbbind(dbproc, #columna, tipo_variable, longitud_variable, variable_local)

donde:

dbproc: Es el apuntador a DBPROCCESS

#columna: Es el número de columna del comando select

Por ejemplo,

select cveproy, nomproy,duracion from proyecto

el número de columna es 1,2 y 3

Si quisiéramos recuperar las mismas columnas usando la siguiente instrucción:

select * from proyecto

entonces el número de columnas debe ser el correspondiente al orden en que están las columnas en la tabla, en este caso son 1,2 y 6.

tipo_variable: Representa el tipo de variable en nuestro programa cliente, no necesariamente tiene que ser el mismo tipo de la tabla.

longitud_variable: Se pone el número 0 para que la variable local sea suficientemente grande para almacenar los resultados generados por la instrucción SQL.

variable_local: Es el nombre de la variable definida en nuestro programa en C.

Para variables de tipo entero se requiere poner el símbolo '&' antes del nombre de la variable.

Por ejemplo,

```
DBCHAR      nomproy[250];       /* cadena */
DBINT       duracion;           /* entero */

select nomproy,duracion from proyecto
dbbind(dbproc,1,NTBSTRINGBIND,0,nomproy);
dbbind(dbproc,2,INTBIND,0,&duración);
```

dbnextrow(). Lee un renglón del resultado del comando SQL y pone el resultado en la variable de programa especificada por *dbbind()*. Cada llamada exitosa de está rutina lee otro renglón hasta que el último renglón sea leído y es regresada el valor NO_MORE_RESULTS. El procesamiento de los resultados toma lugar dentro de un ciclo, porque cada llamada a *dbnextrow()* sobrescribe el valor previo de las variables de programa.

dbexit(). Cierra las conexiones con el servidor SQL y vacía el DBPROCESS. Este también limpia cualquier estructura inicializada por *dbinit()*. **Esta debe ser la última rutina en un programa que use las DB-Library.**

Ejemplo (proyecto.c) usando las DB-Library's

```
/*
**      proyecto.c
**      Este ejemplo muestra el comportamiento básico de muchas aplicaciones con
**      DB-Library's. El programa abre una conexión con el servidor SQL, envía el
**      comando select de SQL al servidor, y procesa el conjunto de renglones
**      resultantes del select. El programa utiliza la tabla "proyecto" de la base
**      de datos "papiit" de Sybase.
*/
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
```

```
main(){
DBPROCESS    *dbproc;              /* La conexión con el servidor SQL */
LOGINREC     *login;               /* Información de nuestra conexión */
RETCODE      return_code;

/*
** Estas variables de programa son para guardar los datos regresados
*/
DBCHAR       cveproy[8];

/*
** Inicializando las DB-Library
*/
if (dbinit() == FAIL)
exit(ERREXIT);

/*
** Creando una estructura LOGINREC y llenándola con información de nuestra
** conexión
*/
login = dblogin();
DBSETLUSER(login, "papiit");          /* Usuario de la base de datos */
DBSETLPWD(login, "papiit@admin");  /* Clave de acceso del usuario */

/*
** Creando la estructura DBPROCESS para la comunicación con el servidor SQL.
** El segundo argumento en la función especifica el nombre del servidor, un valor
** NULL toma como default el nombre del servidor especificado por la variable de
** ambiente DSQUERY (veremos más adelante las variables de ambiente)
*/
dbproc = dbopen(login, NULL);

/*
** Vamos a recuperar los números de proyecto PAPIIT del año 1997.
*/

/*
** Primero, hay que poner el comando en el buffer de comandos.
** Obsérvese el espacio en blanco entre la primera y la segunda instrucción (entre
** las comillas en el segundo renglón y la cláusula where), éste es necesario para
** separar las palabras proyecto y where.
*/
dbcmd(dbproc, "select cveproy from proyecto");
dbcmd(dbproc, " where substring(cveproy,7,2) = '97' ");  /* Aquí hay un espacio en
blanco */

/*
** El servidor SQL procesa el comando en el siguiente orden:
** 1) Va a revisar los errores de sintaxis (por ejemplo, "use database papiit" es
**    sintácticamente incorrecto; éste debería ser "use papiit").
** 2) La segunda revisión una revisión semántica (por ejemplo, "select * from
proyecte"
**    será incorrecto porque la palabra correcta debe ser "proyecto").
```

```
** 3) La tercera revisión ocurre en la actual fase de ejecución. Esta revisión involucra
**    asuntos como permisos o problemas de memoria.
** En la fase de ejecución, dbsqlexec() y dbresults() pueden regresar el valor
** "SUCCEED", el cual significa que el comando tuvo éxito y que existen datos
** para procesar. Un valor de "FAIL" significa que el comando falló y que no hay
** datos para procesar. Un valor de "NO_MORE_RESULTS" significa que no hay más
** datos para procesar. Por lo tanto, el programador debe revisar le valores de
** regreso después de dbsqlexec() y dbresults(), como se muestra a continuación.
*/


/*
** Envía el comando al servidor SQL e inicia su ejecución
*/
if (dbsqlexec(dbproc) == FAIL){
        printf("Error de sintaxis en el comando SQL.\n");
        dbexit();
        exit(ERREXIT);
}


/*
** Procesamiento de resultados
*/
if (dbresults(dbproc) == SUCCEED){
        /* Mapear las columnas a variables */
        dbbind(dbproc, 1, NTBSTRINGBIND, 0,cveproy);

        /* Recuperar e imprimir los renglones */
        printf("Proyectos \n");
        printf("---------- \n");
        while (dbnextrow(dbproc) != NO_MORE_ROWS)
                printf("%s \n", cveproy);
}

/*
** Cerrar nuestra conexión con el servidor y salir del programa
*/
dbexit();
exit(STDEXIT);
}
```

Para verlos resultados que arrojaría nuestro programa proyecto.c, tenemos que revisar el estado de algunas variables de ambiente y después compilarlo para que se pueda ejecutar.

Variables de ambiente

Las variables de ambiente retienen información en la memoria del sistema operativo. Estas pueden ser usadas por cualquier programa.

Al usar las DB-Library se tienen que definir las variables de ambiente SYBASE y DSQUERY como sigue:

SYBASE /opt/sybase Especifica el directorio donde está instalado Sybase
DSQUERY SYBASE Define el nombre del servidor Sybase

Para verificar si existen éstas variables se puede usar el comando "setenv" o "printenv" desde el prompt del sistema operativo UNIX ($setenv o $printenv). En caso de que no existan se pueden declarar desde el mismo prompt del sistema tecleándolas tal y como son definidas como sigue:

$setenv SYBASE /opt/sybase

$setenv DSQUERY SYBASE

O si se prefiere, se pueden definir éstas variables en el archivo **.cshrc** (que es el archivo de configuración de nuestra cuenta en Unix, es equivalente al archivo *autoexec.bat* de una computadora personal), para no tener que definirlas manualmente. El contenido de un programa .cshrc se muestra a continuación:

```
$more .cshrc
# @(#)cshrc 1.11 89/11/29 SMI
umask 022
set path = (/bin /usr/bin /usr/sbin /usr/ucb /opt/cygnus-sol2-1.1/bin /usr/ccs/bin
 /etc /usr/local/bin /opt/sybase/bin /usr/openwin/bin . )
setenv SYBASE /opt/sybase
setenv DSQUERY SYBASE
setenv
LD_LIBRARY_PATH = /usr/openwin/lib:/usr/lib:/opt/SUNWmotif/lib:/opt/SUNWits
/Graphics-sw/xil/lib
set history = 40
```

Una vez que estamos seguros de que existen las variables de ambiente necesarias para acceder a las base de datos Sybase, procedemos a compilar nuestro programa fuente (proyecto.c) y crear nuestro programa ejecutable (proyecto) para desplegar resultados.

Compilación de un programa en C

Para que nuestro programa pueda ser ejecutado desde el sistema operativo, tenemos que realizar un proceso de compilación. La compilación se realiza utilizando la utilería **make** de UNIX.

make lleva automáticamente la cuenta de las dependencias (archivos .h, .c, .o, .lib, etc.) y hace que sea fácil crear programas ejecutables. Al utilizar make se especifica el modo en que las partes de un programa son dependientes de otras partes de código. Esta especificación de las dependencias subadyacentes de un programa se coloca en un archivo *makefile* o *Makefile*. Al ejecutar la orden

```
$ make
```

el programa busca un archivo llamado *makefile* o *Makefile* en el directorio actual. Examina el *makefile*; recompila los archivos fuente que han sido modificados desde que fueron compilados la última vez y también recompila cualquier archivo que dependa de otro que haya sido modificado. Podemos decir entonces que **make** ahorra tiempo en la compilación de programas.

Los comentarios en un archivo *makefile* pueden insertarse utilizando el símbolo #. Todo el texto que haya después del # será ignorado por **make**.

Si no es especificada una etiqueta en **make**, éste toma la primera etiqueta especificada en el archivo *makefile*.

Para compilar nuestro ejemplo anterior **proyecto.c** necesitamos tener el siguiente archivo *makefile*:

```
# Demostración del uso de la herramienta make de UNIX
# para la programación de con las DB-library.
#
# makefile 1.3 5/24/90
#
# Cambia las siguientes definiciones de acuerdo con tus directorios de trabajo
SYBASE    = /opt/sybase
INCDIR    = $(SYBASE)/include
LIBDIR    = $(SYBASE)/lib
HEADERS   = $(INCDIR)/sybfront.h \
            $(INCDIR)/sybdb.h
DBLIBS =  $(LIBDIR)/libsybdb.a
INCLUDE   = -I. -I$(INCDIR)

# Es muy importante separar la etiqueta "proyecto:" de la instrucción $(HEADERS)
proyecto.c
# con un tabulador. La separación con un espacio en blanco causaría un error a la
# hora de la compilación.

proyecto: $(HEADERS) proyecto.c
          gcc $(INCLUDE) proyecto.c $(DBLIBS) -lm -lnsl -o proyecto
```

y teclear el siguiente comando desde el prompt de UNIX:

```
$make proyecto
gcc -I. -I/opt/sybase/include proyecto.c /opt/sybase/lib/libsybdb.a -lm -lnsl -o proyecto
```

El resultado de la compilación sería un programa ejecutable llamado **proyecto** ubicado en el directorio donde se realizó la compilación.


make, como dijimos compila sólo los archivos fuente que hayan sido modificados, por ejemplo, si compilamos nuevamente el programa **proyecto.c** make mandaría el siguiente mensaje `**proyecto is up to date**, lo que indica que el programa no ha tenido modificaciones.

Ahora nosotros estamos en la posibilidad de ejecutar el programa ejecutable llamado **proyecto** desde el indicador del sistema operativo y ver

los resultados en pantalla, como se muestra a continuación:

```
$proyecto <Enter>
Proyectos
------------
IN200697
IN300297
IN200897
IN111797
```

Programación de CGI's

El Common Gateway Interface (CGI) es un estándar para programas que sirven de puente entre el Web y una aplicación.

Por ejemplo, uno puede querer que su aplicación, por lo general de bases de datos, pueda utilizarse a través de páginas del Web. Así, se puede hacer un programa que haga una interfaz para el usuario (forma WWW) quien llena esta solicitud llenando ciertos campos de texto o seleccionando opciones de menús. Esta solicitud es recogida por el CGI quien la transforma en comandos y datos inteligibles por la aplicación, a su vez ésta le regresa al CGI los resultados de la consulta y el CGI se los despliegue al usuarios en una nueva forma WWW.

El estándar CGI surgió al mismo tiempo que el Web, para ampliar sus posibilidades al incluirle programación. Un CGI siempre corre en el servidor, lo que significa que cuando se accede a una página del Web que contiene un CGI, éste se ejecuta en el servidor y lo que se ve posteriormente en una página, es el resultado de esta ejecución. Esto implica lo siguiente:

- Aumento de la carga del servidor, si se ejecutan simultáneamente varios CGIs complejos.
- Por lo anterior, el cliente no sufre sobrecarga.

- Si no se toman las debidas precauciones al crearlo, un CGI puede, en un momento dado, comprometer la seguridad del servidor, puesto que dentro de un CGI se puede ejecutar cualquier programa o desplegar cualquier archivo que se encuentre en dicho servidor.
- Un CGI tiene a su disposición cierta información proveniente del cliente y a través de variables de ambiente se puede conocer, por ejemplo la dirección IP y el tipo de cliente.
- Al igual que casi todo en el Web, los CGIs no pueden funcionar al mismo tiempo con el usuario pues tienen que esperar a que el cliente los mande ejecutar. Esto es, que los programas CGI no pueden actuar en forma interactiva con el usuario, sino que deben esperar a que el usuario oprima un botón de "Aceptar" o "Submit" para ser ejecutados.

Un programa que siga el estándar CGI está escrito en algún lenguaje de programación interpretado o compilado como puede ser C, Perl o scripts de Unix (Shell de Unix) y que manda a ejecutar una aplicación y despliega los resultados en formas WWW.

Los programas CGI tienen que residir bajo el directorio `cgi-bin` (en nuestro caso usamos el directorio `/home/http/httpd_1.4.2/cgi-bin/solicitud`) para que puedan ser ejecutados por el servidor HTTP.

Para que un programa común y corriente se convierta en CGI tiene que generar como resultado un documento HTML. Para ejemplificar, vamos a escribir el tan famoso programa "Hola Mundo" en versión CGI:

```
main(){
        printf("Content-type: text/html%c%c",10,10);
        printf("<H1>Hola Mundo</H1>");
```

La primera línea (no aparece en la página del Web pero hay que ponerla), dice que lo que a continuación viene es un `texto` del tipo `html`. En la segunda línea se pueden observar las etiquetas de HTML para ajustar el tipo de texto (letras grandes, en este caso).

Presentaremos, ahora un ejemplo de una forma de captura. La forma será generada por un programa CGI.

```
/*
** solicitud.c
** Es una forma de captura generada por un CGI.
** Con negritas se ponen las consideraciones para que pueda ser
** visualizada desde Netscape.
** NOTA: argv[1] = IN100197, argv[2] = BOOL570224 y argv[3] = 2
*/
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>

main(){
DBPROCESS    *dbproc;
LOGINREC     *login;
RETCODE      return_code;
/*
** Especificación de un  texto tipo HTML
*/
printf("Content-type: text/html%c%c",10,10);

if (dbinit() == FAIL)
exit(ERREXIT);
login = dblogin();
DBSETLUSER(login, "papiit");
DBSETLPWD(login, "papiit@admin");
dbproc = dbopen(login, NULL);
/*
** Creando la forma WWW de salida (observensen los códigos en HTML
** <FORM>, <TITLE>, etc.)
*/
printf("<FORM METHOD=POST ACTION=/cgi-bin/solicitud/cgisol>\n");
printf("<TITLE>Datos Generales del Proyecto</TITLE>\n");
printf("<CENTER><H3>DATOS                    GENERALES                    DEL
PROYECTO</H3></CENTER>\n");
printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");

printf("<INPUT TYPE=HIDDEN NAME=cveproy      VALUE=\"%s\">\n",argv[1]);
```

```
printf("<INPUT TYPE=HIDDEN NAME=rfc          VALUE=\"%s\">\n",argv[2]);
printf("<INPUT TYPE=HIDDEN NAME=numanio      VALUE=\"%s\">\n",argv[3]);
}
printf("<PRE><H4>\n");

printf("NOMBRE DEL PROYECTO:\n");
        printf("<INPUT TYPE=TEXT    NAME=nomproy  SIZE=60
MAXLENGTH=250><P>\n");
dbfcmd(dbproc,"select * from disciplina order by nomdis");
if (dbsqlexec(dbproc) == FAIL){
     printf("Error de acceso al servidor\n");
     dbexit();
     exit(ERREXIT);
}
dbresults(dbproc);
dbbind(dbproc,1,NTBSTRINGBIND,0,cvedis);
dbbind(dbproc,2,NTBSTRINGBIND,0,nomdis);
printf("DISCIPLINA:\n");
printf("<SELECT NAME=disciplina SIZE=4>\n");
while(dbnextrow(dbproc) != NO_MORE_ROWS)
        printf("<OPTION>%s\n", nomdis);
printf("</SELECT><P>\n");
printf("DURACIÓN DEL PROYECTO: <SELECT NAME=duracion>");
printf("<OPTION>1\n");
printf("<OPTION>2\n");
printf("<OPTION>3\n");
printf("</SELECT> Año(s)<P>");

printf("</H4></PRE>\n");

printf("<CENTER>Para salvar la información capturada presione aquí: <P>\n");
printf("<INPUT TYPE=SUBMIT VALUE=Aceptar></CENTER><P>\n");
printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");


printf("</FORM>\n");
dbexit();
exit(STDEXIT);
}
```

Téngase atención en las siguientes líneas:

```
/* Especifica que el  texto es de tipo HTML */
printf("Content-type: text/html%c%c",10,10);


/* Creación de una forma WWW de presentación de datos */
```

printf("<FORM METHOD=POST ACTION=/cgi-bin/solicitud/cgisol>\n"); Abre la forma

printf("</FORM>\n"); Cierra la forma

Para poder correr el programa desde Netscape, tenemos que generar un programa ejecutable de éste.

El archivo **solicitud.c** debe pasar por los siguientes pasos para ser un programa ejecutable:

a) Incluir las siguientes líneas en el archivo makefile

```
# Demostración del uso de la herramienta make de UNIX
# Es muy importante separar la etiqueta (solicitud) de la instrucción $(HEADERS)
# con un tabulador. La separación con un espacio en blanco causaría un error a la
# hora de la compilación.
solicitud: $(HEADERS) solicitud.c
        gcc $(INCLUDE) solicitud.c $(DBLIBS) -lm -lnsl -o solicitud
```

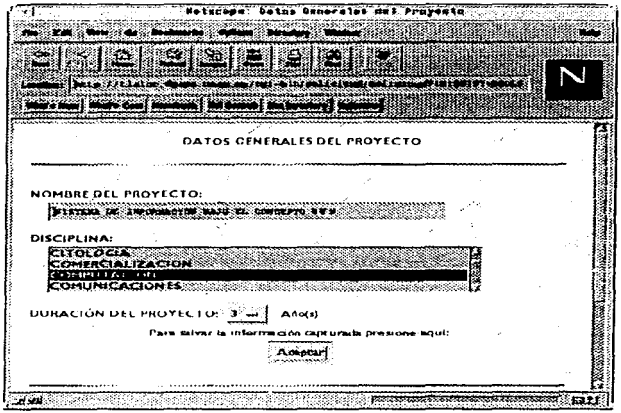
b) Compilar el archivo solicitud.c con la herramienta make de Unix (para crear el programa ejecutable):

```
$make solicitud
```

c) Copiar el programa ejecutable **solicitud** al directorio /home/http/httpd_1.4.2/cgi-bin/solicitud

Al correr el programa desde la opción del menú "File -> Open Location" de Netscape:

Se vería lo siguiente:



Un gran camino para aprender Formas WWW es examinando páginas en el Web: puedes ver el código fuente de las que te hayan impresionado. Usa la opción "Document Source" del menú "View" de tu visualizador, o la tecla "\" en el caso del visualizador lynx.

Los CGI utilizan variables de ambiente para acceder los datos capturados en una forma WWW. A continuación veremos algo sobre variables de ambiente.

Variables de ambiente

Las variables de ambiente son cadenas de texto (parejas de nombre y valor) que son usadas programas shell de Unix u otros programas (algunos programas pueden fijar éstas), y las cuales pueden ser accesadas por otros programas que se estén ejecutando. Este es un simple camino para programas (como son el shell o, en este caso, al servidor HTTP o servidor

Web) para pasar datos a otros programas. Estas cadenas de texto son llamadas "variables de ambiente" porque ellas son globales, y accesibles por todos los programas que se estén ejecutando.

Para ver que variables están disponibles en los programas CGI, nosotros vamos a escribir un sencillo programa en shell de Unix. Aquí está el script:

```
echo Content-type: text/plain
echo

echo Test de variables de ambiente disponibles para un programa CGI
echo
echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = $PATH_INFO
echo PATH_TRANSLATED = $PATH_TRANSLATED
echo SCRIPT_NAME = $SCRIPT_NAME
echo QUERY_STRING = $QUERY_STRING
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```

Guardamos el script en el archivo `variables`, le cambiamos los permisos para ejecución (**$chmod 755 variables**), copiamos el archivo al directorio de trabajo `/home/http/httpd_1.4.2/cgi-bin/solicitud` y lo ejecutamos desde la opción del menú "File -> Open Location" de Netscape (**tlaloc.dgapa.unam.mx/solicitud/variables**). El resultado sería el siguiente:

```
Test de variables de ambiente disponibles para un programa CGI

SERVER_SOFTWARE = NCSA/1.4.2
SERVER_NAME = tlaloc.dgapa.unam.mx
GATEWAY_INTERFACE = CGI/1.1
```

```
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/solicitud/var
QUERY_STRING =
REMOTE_HOST = tlaloc.dgapa.unam.mx
REMOTE_ADDR = 132.248.37.2
REMOTE_USER = 011
CONTENT_TYPE =
CONTENT_LENGTH =
```

Nota que cada línea de una variable de ambiente consiste de un nombre (usualmente en mayúsculas), un signo de '=', y luego el valor de la variable.

Aquí hay algunas de las variables de ambiente más usadas en los scripts y programas.

- **SERVER_PORT**: Es el número de puerto donde la petición fue enviada.
- **REQUEST_MODE**: Es el método utilizado por la petición, comúnmente es GET o POST .
- **SCRIPT_NAME**: Es la ruta donde se encuentra el programa que se esta ejecutando.
- **QUERY_STRING**: Contiene información de una forma que usa el método GET.
- **CONTENT_TYPE**: Contiene el tipo de datos pasados al CGI. Generalmente, éstos van a ser "application/x-www-form-urlencoded", el cual es el tipo de datos pasados desde una forma HTML que usa el método POST.

CONTENT_LENGTH: Es el número de caracteres (o bytes) de los datos pasados al CGI.

Más adelante veremos los métodos POST y GET.

Para accesar una variable de ambiente desde un script en shell, simplemente precedimos ésta con el carácter '$', por ejemplo:

```
$REMOTE_HOST
```

Para accesar una variable desde un programa en C, usamos la función *getenv()* (definida en *stdlib.h*). Por ejemplo:

```
if (strcmp(getenv("REQUEST_METHOD"),"POST")){
    printf("Este script debe referenciarse con un metodo POST.\n");
    exit(1);
}
if (strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")){
   printf("Este programa solo puede ser usado para decodificar datos de una forma.
\n");
  exit(1);
}
```

Recuperación de datos de una forma WWW

Algo que nos preguntaríamos, es *Cómo vamos a recuperar los datos introducidos por el usuario en una forma?*, la respuesta es la siguiente: existen dos métodos los cuales pueden ser usados para acceder la información de las formas. Estos métodos son GET y POST. Dependiendo de cual método tu uses, tu vas a recibir los resultados de la forma de diferentes manera.

El método GET

Normalmente, GET es usado para conseguir un archivo u otro recurso, posiblemente con parámetros especificando más exactamente que es necesitado.

En caso de una forma de captura, GET incluye completamente la entrada en el URL, así:

http://tlaloc.dgapa.unam.mx/solicitud/solicitud?**name1 = value1&name2 = value2**

Este también puede ser usado para muchas formas de captura, si estas no tienen muchos datos (el límite varia de visualizador a visualizador).
Con GET el programa CGI recibirá los datos de la forma en la variable de ambiente **QUERY_STRING**.

El método POST
Normalmente, POST es usado para enviar gran cantidad de datos a un servidor para ser procesados (El nombre de POST probablemente viene de la idea de enviar una nota a un grupo de discusión). Cuando una forma HTML es enviada usando POST, tus datos son adheridos al final de la petición POST, en su propio objeto. También el tamaño de los datos en POST no es limitado como en GET.
Con POST el programa CGI recibirá los datos de la forma desde la entrada estándar (stdin). El servidor no enviará un identificador (EOF) al final de los datos, en su lugar tu debes usar la variable de ambiente **CONTENT_LENGTH** para determinar cuantos datos deberías leer de la entrada estándar.

Las ventajas de POST son que tu no estas limitado en el envío de datos y que tu puedes contar con tu script cada vez que la forma es enviada. La ventaja de GET es que toda la forma enviada puede ser encapsulada en un URL.

 En nuestros ejemplos con formas utilizaremos el método POST.

Decodificación de formas

Los datos de una forma HTML son enviados a un programa CGI en pares de *NAME/VALUE* (nombre/valor) en el siguiente formato:

```
NAME1=VALUE1&NAME2=VALUE2&NAME3=VALUE3& etc.
```

Para decodificar estos datos, un programa CGI tiene que dividir la cadena en campos de *NAME* y *VALUE*. Esto es hecho buscando el siguiente carácter en la entrada:

= representa el final del nombre (*NAME*) de una variable en una forma. & representa el final del valor (*VALUE*) de una variable en una forma.

Una vez que han sido separados los pares de *NAME* y *VALUE* los caracteres especiales en la entrada deben ser convertidos en sus valores reales ASCII. Estos caracteres especiales son los siguientes:

Carácter Introducido	Carácter enviado al CGI
espacio	+
=	%3D
Return	%0A
%	%25
&	%38

La conversión tiene que ser realizada tanto en el nombre (*NAME*) como en el valor (*VALUE*) de la variable. Esto es especialmente importante en campos de texto, donde cualquier carácter ASCII puede ser introducido por el usuario.

Del ejemplo **solicitud** visto anteriormente, estas eran los campos de entrada que se pedían :

```
/* NOTA: argv[1] = IN100197, argv[2] = BOOL570224 y argv[3] = 2 */

printf(" <INPUT TYPE = HIDDEN NAME = cveproy      VALUE = \" %s\" >\n",argv[1]);
printf(" <INPUT TYPE = HIDDEN NAME = rfc          VALUE = \" %s\" >\n",argv[2]);
printf(" <INPUT TYPE = HIDDEN NAME = numanio      VALUE = \" %s\" >\n",argv[3]);
printf("NOMBRE DEL PROYECTO:\n");
printf(" <INPUT TYPE = TEXT  NAME = nomproy  SIZE = 60
MAXLENGTH = 250 > <P>\n");
printf("DISCIPLINA:\n");
printf(" < SELECT NAME = disciplina SIZE = 4 >\n");
printf("DURACIÓN DEL PROYECTO:  < SELECT NAME = duracion > ");
```


Los datos de tipo HIDDEN no se muestran en pantalla, sin embargo, si tienen una valor y se toman como una entrada más.

Al capturar los datos en la forma, tendríamos los siguientes pares de datos:

- **cveproy** = IN100197
- **rfc** = BOOL570224
- **numanio** = 2
- **nomproy** = SISTEMA DE INFORMACIÓN BAJO EL CONCEPTO WWW
- **disciplina** = COMPUTACION
- **duracion** = 3

De acuerdo a los datos capturados, la cadena que recibiría nuestro programa CGI sería la siguiente:

cveproy = IN100197&**rfc** = BOOL570224&**numanio** = 2&**nomproy** = SISTEMA + DE + INFORMA CIÓN + BAJO + EL + CONCEPTO + WWW&**disciplina** = COMPUTACION&**duracion** = 3

La longitud de la cadena anterior es de 127 caracteres, por lo tanto la variable CONTENT_LENGTH tendrá un valor igual.

En el programa CGI vamos a determinar la longitud de la variable CONTENT_LENGTH con la siguiente instrucción:

```
/* Determinar la longitud de la cadena que va a decodificar el CGI */
cl = atoi(getenv("CONTENT_LENGTH"));
```

Después vamos a separar en pares de valores con las siguientes instrucciones:

```
/* Separar en pares de datos de la forma NAME=VALUE
** entries[x].val contiene los valores de las variables
** entries[x].name contiene los nombre de las variables
*/
/* Se leerán todos datos de la entrada estándar hasta que el número de
   estos sea igual a el valor de la variable CONTENT_LENGTH */
for (x=0;cl && (!feof(stdin));x++) {
     /* m contiene el número de datos capturados en la forma menos uno   */
     /* esto se debe a que se empieza a contar desde 0                    */
     m=x;

     /* Separar en pares de datos (la referencia es el símbolo '&') */
     entries[x].val = fmakeword(stdin,'&',&cl);

     /* Cambiar los símbolos '+' por espacios en blanco */
     plustospace(entries[x].val);

    /* Eliminar los caracteres de escape  (como return's) */
     unescape_url(entries[x].val);

    /* Separar los pares de datos en NAME y VALUE (la referencia es el símbolo '=')
*/
     entries[x].name = makeword(entries[x].val,'=');
}
```

Entonces, para nuestro ejemplo:

```
x=0
m=0
entries[0].val   = cveproy=IN100197
entries[0].val   = cveproy=IN100197
entries[0].val   = cveproy=IN100197
entries[0].val   = IN100197
entries[0].name          = cveproy
```

```
...
x = 3
m = 3
entries[3].val   =
nomproy = SISTEMA + DE + INFORMACIÓN + BAJO + EL + CONCEPTO + WWW
entries[3].val   = nomproy = SISTEMA DE INFORMACIÓN BAJO EL CONCEPTO WWW
entries[3].val   = nomproy = SISTEMA DE INFORMACIÓN BAJO EL CONCEPTO WWW
entries[3].val   = SISTEMA DE INFORMACIÓN BAJO EL CONCEPTO WWW
entries[3].name          = nomproy
...
x = 5
m = 5
```

El número de datos leídos son 6 (m = 5, porque el conteo inicia desde el dato 0,1,2,3,4,5 que en total son 6 datos).

Las funciones utilizadas para decodificar los datos se encuentran definidas en el apéndice C en el archivo **util.c**, y en lo sucesivo se usará sólo el archivo objeto **util.o** resultado de compilar el archivo **util.c** con la siguiente instrucción **$gcc -c util.c**.

Las estaciones de trabajo permiten seleccionar y deseleccionar una opción de un menú, por ejemplo, en la solicitud del ejemplo antes visto se puede no seleccionar la disciplina del menú de disciplina.

Los valores que no se capturan en las cajas de texto son identificados como nulos (NULL) por los CGIs.

La variable **m** nos servirá para hacer validaciones sobre los datos hayan introducido, por ejemplo, podemos ver si se capturó la disciplina en la solicitud, de la siguiente manera:

```
/* Si m no es 5, no se capturó la disciplina */
if (m != 5){
    printf("<H3>Favor de seleccionar la disciplina </H3><P>\n");
            printf("Utilice  <B>Back\</B>  del  menú  de  Netscape  para
seleccionarla.<P>\n");
    exit(1);
}
```

Adicionalmente, nosotros podemos convertir a letras mayúsculas los datos capturados por el usuario, esto es conveniente cuando queremos que tener un estándar en los datos capturados, aunque no siempre es conveniente hacer esto, por ejemplo, en un campo de tipo texto libre (*TEXTAREA*) el usuario puede combinar letras minúsculas y mayúsculas para hacer énfasis en algo. De cualquier manera, mostramos cual es el camino para realizar la conversión de los datos a letras mayúsculas:

```
/*
** Conversión a mayúsculas de los datos capturados en la forma WWW
** (los datos se guardarán en la base de datos con letras mayúsculas)
*/
  for(x = 0; x <= m; x++){
     nom = entries[x].val;
     if(x! = 8){
        while (*nom! = '\0'){
           *nom = toupper(*nom);
           nom++;
        }
        while (*entries[x].val == ' '){
           entries[x].val = entries[x].val++;
           entries[x].val++;
        }
     }
  }
```

Vamos a crear un programa CGI que decodificará los datos capturados en la forma de solicitud que presentamos anteriormente y posteriormente el CGI dará una respuesta a través de una forma en HTML.

Recuérdese la línea del programa **solicitud.c**:

```
printf(" <FORM METHOD = POST ACTION = /cgi-bin/solicitud/cgisol>\n");
```

En está línea especificamos que será el programa **cgisol** el que decodificará los datos capturados y el que dará una respuesta. También, decimos que se usará el método POST para la Decodificación de los datos.

Recomendamos que al declarar variables involucradas con campos de tablas se usen los mismos nombres de los campos, por ejemplo, para mapear la columna de la clave de la disciplina (cvedis) se use una la del cvedis del tipo DBCHAR (DBCHAR cvedis[5];). Lo anterior es para tener un mejor manejo de las variables.

```
/*
** cgisol.c
** Programa que decodifica la forma generada por el ejemplo solicitud.c
** Utiliza el método POST para decodificar los datos.
*/
#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <string.h>

#define MAX_ENTRIES 10000

/*
** Definición de la estructura de datos que va a guardar los valores
** capturados en una forma WWW
*/
typedef struct {
   char *name;
   char *val;
} entry;

/*
** Declaración de las funciones que decodifican una forma WWW
** (su definición está en el archivo util.c)
*/
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);


main(int argc, char *argv[]) {
   /*
   ** Declaración de variables
   */
   entry entries[MAX_ENTRIES];
   register int x,m=0;
   int cl,dur,band;
```

```
DBPROCESS   *dbproc;
LOGINREC    *login;
RETCODE      return_code;
char *nom;

DBCHAR      cvedis[5];

/*
** Especificación de un  texto tipo HTML
*/
printf("Content-type: text/html%c%c",10,10);

/*
** Verificación del documento:
** debe tratarse de una forma WWW (application/x-www-form-urlencoded)
** y ser referenciada a través del método POST (ACTION=POST)
** para poder ser interpretado el documento por el programa CGI.
*/
if (strcmp(getenv("REQUEST_METHOD"),"POST")){
   printf("Este script debe referenciarse con un metodo POST.\n");
   exit(1);
}
if (strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")){
   printf("Este programa solo puede ser usado para decodificar datos de una forma. \n");
   exit(1);
}
/* Determinar la longitud de la cadena que va a decodificar el CGI */
cl = atoi(getenv("CONTENT_LENGTH"));

/*
** Decodificación de la forma:
** los valores se guardarán uno a uno de la siguiente manera:
** m tendrá el número de campos existentes en la forma
** entries[x].val alojará los valores de los campos capturados,
** entries[x].name alojará el nombre de los campos de la  forma
** (incluyendo los tipo HIDDEN); x=0,...,m-1
*/
for (x=0;cl && (!feof(stdin));x++) {
   m=x;
   entries[x].val = fmakeword(stdin,'&',&cl);
   plustospace(entries[x].val);
   unescape_url(entries[x].val);
   entries[x].name = makeword(entries[x].val,'=');
}

/*
** Conexión con la base de datos Sybase
*/
if (dbinit() == FAIL)
   exit(ERREXIT);
login = dblogin();


DBSETLUSER(login,"papiit");
```

```
DBSETLPWD(login,"papiit@admin");
dbproc = dbopen(login,NULL);
if (dbproc == NULL){
   printf("<H1>El servidor no se encuentra arrriba</H1>");
   exit(ERREXIT);
}
dbuse(dbproc,"papiit");

/*
** Cabecera del programa
*/

printf("<BODY
BACKGROUND=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");
printf("<TITLE>Datos Generales del Proyecto</TITLE>\n");
printf("<CENTER><H3>DATOS GENERALES DEL PROYECTO</H3></CENTER>\n");
printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");

/*
** Verificando que se hayan capturado todos los datos pedidos
** en la forma
*/
if (*entries[5].val == NULL){
   printf("<h3>Favor de escribir el nombre del proyecto</h3><P>\n");
   printf("Utilice <B>Back</B> del menú de Netscape para seleccionarla.<P>\n");
   exit(1);
}
if (m != 5){ /* Si m no es 5, no se capturó la disciplina */
   printf("<H3>Favor de seleccionar la disciplina </H3><P>\n");
   printf("Utilice <B>Back</B> del menú de Netscape para seleccionarla.<P>\n");
   exit(1);
}

/*
** Conversión a mayúsculas de los datos capturados en la forma WWW
** (los datos se guardarán en la base de datos con letras mayúsculas)
*/
for(x=0; x <= m; x++){
   nom=entries[x].val;
   if(x!=8){
      while (*nom!='\0'){
         *nom=toupper(*nom);
         nom++;
      }
      while (*entries[x].val == ' '){
         entries[x].val=entries[x].val++;
         entries[x].val++;
      }
   }
}

/*
```

```
   ** Actualización de la base de datos
   ** En este caso se realizará una instrucción update, pero puede ejecutarse
   ** cualquier otro instrucción como insert o delete.
   */
    dbfcmd (dbproc,"select cvedis from disciplina
                        where nomdis = \"%s\" ",
                        entries[4].val);          /* Recuperamos primero la clave de la
disciplina */
   if (dbsqlexec(dbproc) == FAIL){
      printf("<H2>Error durante el proceso de acceso al servidor</H2>");
      dbexit();
      exit(ERREXIT);
   }
   dbresults(dbproc);
   dbbind(dbproc,1,NTBSTRINGBIND,0,cvedis);
   dbnextrow(dbproc);
   dbcanquery(dbproc);
   dbfcmd(dbproc,"update proyecto set
                        nomproy = '%s',
                        cvedis = '%s',
                        duracion = %s
                     where
                        cveproy = '%s' ",
                        entries[3].val,
                        cvedis,
                        entries[5].val,
                        entries[0].val);
   if (dbsqlexec(dbproc) == FAIL){
      printf("<H2>Error durante el proceso de acceso al servidor</H2>");
      dbexit();
      exit(ERREXIT);
   }
   /*
   ** Respuesta del programa CGI a través de una forma WWW
   */
   printf("<H3>Los datos generales del proyecto son los siguientes:</H3>\n");
   printf("<H4>\n");
   printf("Clave del Proyecto: <B>%s</B><P>\n",entries[0].val);
   printf("Nombre del Proyecto: <B>%s</B><P> \n", entries[3].val);
   printf("Disciplina: <B>%s</B><P>\n",entries[4].val);
   printf("Duración del proyecto: <B>%s años(s)</B><P>\n",entries[5].val);
   printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");

   /*
   ** Pie del programa
   */
   printf("</BODY>\n");
   printf("</FORM>\n");
   dbexit();
   exit(ERREXIT);
}
```

Para que el programa CGI pueda decodificar la forma debe crearse el programa ejecutable de éste mediante los siguientes pasos:

El archivo **cgisol.c** debe pasar por los siguientes pasos para ser un programa ejecutable:

a) Incluir las siguientes líneas en el archivo makefile

```
# Demostración del uso de la herramienta make de UNIX
# Es muy importante separar la etiqueta (cgisol) de la instrucción $(HEADERS)
# con un tabulador. La separación con un espacio en blanco causaría un error a la
# hora de la compilación.
# util.o es el archivo que tiene las definiciones de las funciones que decodifican una forma
cgisol: $(HEADERS) cgisol.c
        gcc $(INCLUDE) cgisol.c  util.o $(DBLIBS) -lm -lnsl -o cgisol
```

b) Compilar el archivo cgisol.c con la herramienta make de Unix (para crear el programa ejecutable):

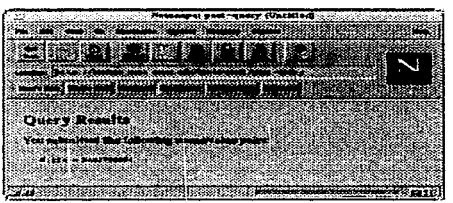
```
$make cgisol
```

c) Copiar el programa ejecutable **cgisol** al directorio /home/http/httpd_1.4.2/cgi-bin/solicitud

Recomendación: Cuando vayas a escribir un programa CGI, antes de que lo hagas utiliza el programa post-query (visto al inicio) para que veas que pares de datos recibirla el programa CGI. Recordando, el programa anteriormente visto:

```
<FORM METHOD = POST ACTION = "http://hoohoo.ncsa.uiuc.edu/htbin-post/post-query" >
Introduzca su RFC: <INPUT TYPE = text NAME = rfc SIZE = 10 VALUE = "" >
<BR >
<INPUT TYPE = reset VALUE = Deshacer >
<INPUT TYPE = submit VALUE = Aceptar >
</FORM >
```

Estas serían los pares de datos:



Manejo de errores de programación en Netscape

En la programación bajo WWW frecuentemente nos encontramos con distintos errores, que en muchos casos no sabemos ni por que se generan. Al error provocado por una instrucción incorrecta en un programa Netscape lo reconoce como un **Error 500.**

Es importante notar que este tipo de errores no son detectados por el compilador, pues estos errores se generan en el tiempo de ejecución del programa. A continuación veremos los errores de programación más frecuentes que causan dicho error.


Los ejemplos mostrados están escritos en lenguaje C.

Uso de un argumento que no existe

Por ejemplo, nosotros asumimos que el usuario tiene que capturar 3 datos en una forma, y no validamos que en realidad haya capturado los 3 datos, entonces vamos a tratar de usar un dato que no existe. Supongamos que capturó sólo 2 datos el usuario, en nuestro programa trataríamos de imprimir lo siguiente:

```
Resultado de la Captura:
Nombre del proyecto = entries[0].val
Duración del proyecto = entries[1].val
Disciplina = entries[2].val          // No se capturó la
disciplina
```

El error se causaría al tratar de usar la variable entries[2].val, la cual no existe porque no se capturó.

Incongruencia de tipos de datos

a) Al desplegar variables en pantalla

Los programas con frecuencia tienen muchas variables, por lo tanto podemos olvidar cual es el tipo de una variable, o simplemente equivocarnos al escribirlas.

Este ejemplo,

```
char caa[2][20];
/* caa[0] = "FISICO-MATEMATICAS";
   caa[1] = "BIOLOGIA"; */
for(i = 0; i < 2; i + + )
      printf("<OPTION VALUE = %d> %s\n", caa[i], i + 1);
```

causaría un error al tratar de desplegar una variable del tipo cadena como un número. Pues como se puede ver los datos están cambiados en la última línea, la manera correcta de escribir ésta línea es la siguiente:

```
printf("<OPTION VALUE = %d> %s\n", i + 1, caa[i]);
```

b) Al usar una tipo de variable no válido en una instrucción

Este ejemplo,

```
char siga;
siga = '1';
if(siga)                    /* if sólo compara valores numéricos */
```

causaría un error al tratar de evaluar la función if un carácter. La corrección de este ejemplo, sería esta:

```
int siga;
siga = 1;
if(siga)                    /* if sólo compara valores numéricos */
```

Inicialización incorrecta de una variable

En un programa, nosotros primero tenemos que declarar las variables que usamos a usar y después inicializarlas (asignarles un valor); sin embargo, podemos hacerlo de una manera incorrecta.

Este ejemplo,

```
char x[3];    /* caracteres */
x[0] = 5;     /* entero */
```

causaría un error al tratar de asignar el número 5 en un arreglo de caracteres. Una versión corregida sería la siguiente:

```
char x[3];    /* caracteres */
x[0] = '5';   /* carácter */
```

Uso de variables no inicializadas (sin valor)

Nosotros podemos declarar las variables que vamos a usar en nuestro programa, pero podemos olvidar inicializarlas (asignarles un valor).

Este ejemplo,

```
DBPROCESS   *dbproc, dbproc2*;          /* Declaramos variables */
if (dbinit() == FAIL)
   exit(ERREXIT);
 login = dblogin();
 DBSETLUSER(login,"papiit");
 DBSETLPWD(login,"papiit@admin");
 dbproc = dbopen(login,NULL);           /* Inicializamos dbproc */
 if (dbproc == NULL){
   exit(ERREXIT);
 }
```

dbfcmd(**dbproc2**,"delete proyecto where cveproy='%s'", entries[0].val);

causaría un error, al tratar de usar la variable **dbproc2** sin antes inicializarla. Tendríamos que agregar las siguientes líneas, entes de ejecutar la instrucción *dbfcmd()*, para corregir este error:

```
dbproc2 = dbopen(login,NULL);                    /* Inicializamos dbproc2 */
  if (dbproc2 = = NULL){
    exit(ERREXIT);
  }
```

Uso incorrecto de funciones de programación

Si no se conocen bien las funciones de un lenguaje de programación, éstas se podrían usar incorrectamente.

a) Funciones

Este ejemplo,

```
char fuente[4] = "juan";
char destino[4] = "";
strcpy(destino,toupper(fuente));
```

causaría un error al tratar de convertir a letras mayúsculas una cadena con la instrucción *toupper()*, y es que ésta función efectivamente realiza una conversión a mayúsculas, sólo que lo hace carácter por carácter y no por cadena. La versión corregida de este ejemplo sería:

```
char fuente[4] = "juan";
char destino[4] = "";
for(int j=0; j < strlen(fuente); j++)
strcpy(destino[j],toupper(fuente[j]));
```

Uso incorrecto de operadores

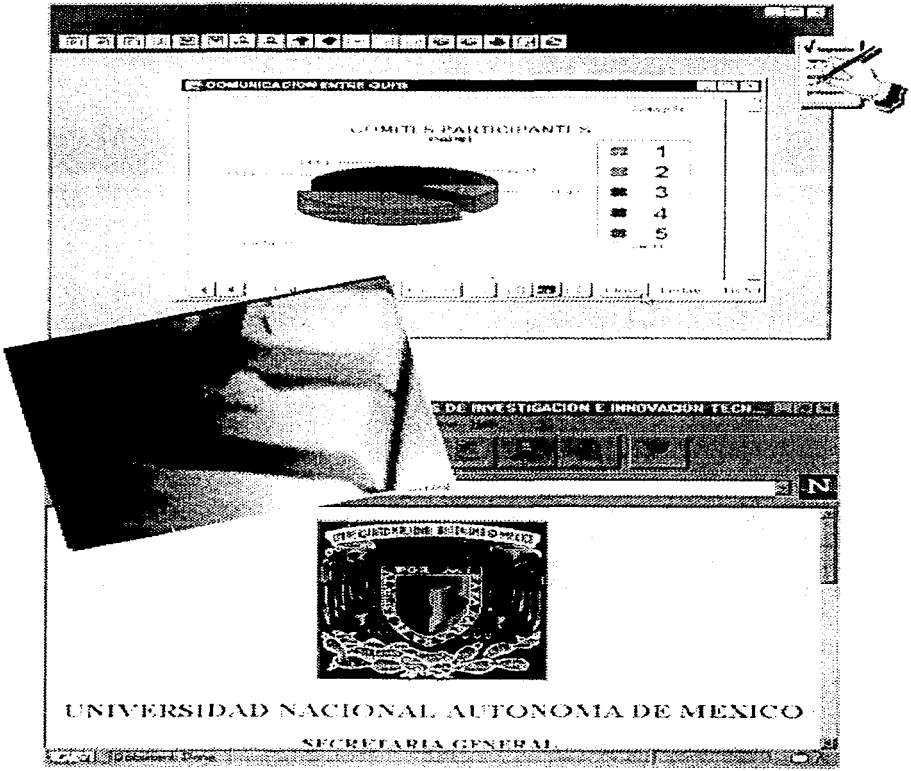
En el uso de apuntadores a cadenas en C se debe considerar lo siguiente:

```
char *texto;                                     /* Incorrecto */
strcpy(texto,"PRUEBA EN NETSCAPE");

char texto[80];                                  /* Correcto */
strcpy(texto,"PRUEBA EN NETSCAPE");
```

Uso de una columna de una tabla o una tabla no existente

En ocasiones, una vez realizado el programa se decide cambiar el la estructura de una tabla dentro de la base de datos. Este cambio, alteraría el funcionamiento del sistema, por lo que cada vez que se modifique una columna de una tabla o una tabla completa se debe reflejar el cambio también en el programa.

Aplicaciones



Análisis y creación de la base de datos

Se realizó el analisis de todos los requerimientos del sistema apoyándonos de entrevistas con el personal administrativo encargado de la manipulación de la información así como también de entrevistas con algunos investigadores, con la finalidad de cubrir sus necesidades de manera más precisa.

El análisis de los requerimientos nos llevaron a identificar a cada una de las entidades, atributos, relaciones entre entidades e identificadores únicos de cada entidad conformando así el diagrama entidad relación. (VER ANEXO D)

En esta fase de diseño se homologaron los atributos y se definieron algunos atributos de uso exclusivo para la manipulación interna de la información . Una vez terminada esta fase procedimos a transcribir nuestro diagrama entidad relación a tablas que conformarian la base de datos del sistema definiendo llaves primarias, llaves foráneas, información contenida en catálogos, así como en las propias tablas, no sin realizar pruebas de escritorio para verificar que las relaciones entre las tablas nos arrojarían la información necesaria.

Concluído este punto continuamos con el diseño físico de la base de datos lo cual conlleva a la asignación de espacio físico en el servidor SYBASE en el que se montara la base de datos, lo cual incluye dispositivos, base de datos y transaction log.

Un dispositivo es un archivo del sistema operativo en el cual son almacenados la base de datos y el log de transacciones, el log de transacciones es una área reservada por SQL Server para llevar un registro de los cambios que son realizados a la base de datos, cada base de datos posee un log de transacciones. Dada la importancia de el log de transacciones se recomienda tener dos dispositivos uno para datos y otro especialmente para el log de transacciones, esto con la finalidad de poder recuperar la información de la base

de datos de manera más sencilla en caso de que ocurriera alguna falla o pérdida de información.

Para la base de datos del sistema (PAPIIT) se definieron dos dispositivos uno para datos de 40 MGB y otro para el log de 10 MGB; esto de a través de la instrucción:

```
DISK INIT
        NAME ='logical _name'
        PHYSNAME ='physical _name'
        VDEVNO =  virutal_device_number
        SIZE  = number_of_2k_bloks
```

describiendo brevemente cada uno de los componentes:

DISK INIT : Proporciona el nombre al dispositivo lógico y físico, preparando el dispositivo en donde se almacenará la base de datos.

NAME : Nombre lógico del dispositivo de la base de datos
PHYSNAME: Nombre físico incluyendo el path del dispositivo de la base de datos

VDEVNO : Dispositivo virtual que identifica el dispositivo de la base de datos. Se pueden usar de 0 a 255 excepto el 0 y el 1 que están asignados para el sistema.

Creación del dispositivo de Datos:

```
DISK INIT
        NAME = ' dev1 '
        PHYSNAME=  ' .\wroot\.....\dev1.dat' ,
        VDEVNO = 2,
        SIZE = 40960
```

dispositivo para el log de transacciones:

```
DISK INIT
        NAME = ' dev2'
        PHYSNAME=  ' .\wroot\.....\dev1.dat' ,
        VDEVNO = 3
        SIZE =  10240
```

Una vez creados los dispositivos se creó la base de datos del sistema:

```
CREATE TABLE PAPIIT
ON dev1 =40
LOG ON dev2 =10
```

Para la creación y borrado de las tablas y vistas, se realizaron scripts en SQL para realizar modificaciones en la base de datos de manera más práctica y fácil, estos scripts también se utilizaron para la inserción de los datos a las tablas. A continuación se muestra parte del contenido del script para crear , borrar e insertar información a las tablas.

Script para crear tablas:

```
print "USANDO PAPIIT..."
use papiit
go
print"TABLA COMITE "
create table comite(cvecom char(2) NULL,nomcom char(50) NULL,maximo tinyint NULL)
go
print "TABLA PROTOCOLO"
create table protocolo5 (cveproy char(8) NULL,antecedentes text NULL,
                objetivos text NULL,hipotesis text NULL,
                metas text NULL,metodologia text NULL,
                infraestructura text NULL)
go
print"TABLA PROYECTO "
create table proyecto6 (cveproy char(8) NULL,nomproy char(250) NULL,
              cvedis char(4) NULL,cvecom char(2) NULL,
             cvedep char(5) NULL,duracion tinyint NULL,
             inicio tinyint NULL)
go
...
```

Scrip para insertar datos las tablas :

```
print "USANDO PAPIIT"
use papiit
go
print "INSERTA EN LA TABLA COMITE"
insert into comite values("1","Ciencias Fisico-Matemáticas y de las Ingenierías","2")
insert into comite values("2","Ciencias Biológicas y de la salud","3")
insert into comite values("3","Ciencias Sociales","2")
insert into comite values("4","Humanidades y Artes","3")
insert into comite values("5","Innovación Tecnológica","2")

print "INSERTA EN LA TABLA PROTOCOLO"
insert into protocolo values ("IN111194","La estructura molecular de las encimas del frijol presentan rugosidad en la
primera etapa de crecimiento", "Proporcionar datos fidedignos que comprueben que esto se debe a que la semilla del
frijol no se encuentra en las condiciones adecuadas para un crecimiento adecuado", "La rugosidad presentada se
debe a que la tierra contiene una cantidad menor de proteínas","Deteminar la cantidad adecuada de proteínas que debe
contener la tierra para evitar esta rugosidad","1.Observar al frijol desdes que es plantado 2.Plantar frjol con las mismas
```

caracteristicas en diferentes tierras cada una con diferentes niveles de proteinas 3. Medir el cremiento por dia de la planta del frijol, 3.Determinar en que dia comienzan aparecer la rugosidad en el frijol","Se cuebta con un invernadero con las condiciones adecuadas, asi como con todo el material necesario para el desarrollo del proyecto,")

```
go
print "INSERTA EN LA TABLA PROYECTO"
cveproy,nomproy,cvedis,cvecom,cvedep,duracion,inicio
insert into proyecto values("IN111194","Rugosidad en el frijol","0200","2","47301","3","94")
insert into proyecto values("IN200697","Las amibas en los vegetales"0100","1","48101","3","97"
insert into proyecto values("IN300297","Los virus informáticos","01004","3","42901","3","97"
insert into proyecto values("IN200897","Cantidad proteinica en el trigo","0200","5","33901","3","97"
go
....
```

Script para el borrado de tablas

```
print "USANDO PAPIIT..."
use papiit
print "TABLA COMITE"
drop table comite
go
print "TABLA PROTOCOLO"
drop table protocolo
go
print"TABLA PROYECTO"
drop table proyecto
go
....
```

Para la inserción de la información proporcionada por el departamento a cargo del manejo administrativo del PAPIIT, éstos nos proporcionaron la información en archivos en dBase, los cuales fueron transformados a archivos de texto como el que se presenta a continuación:

Tabla del proyecto:

```
IN111194,Rugosidad en el frijol,1412,02,1408,3,94
IN200697,Las amibas en los vegetales,1412,02,1408,3,92
IN300297,Aminoacidos,1412,02,1408,3,93
IN200897,Cantidad proteinica en el trigo,2,02,1408,3,94
IN111595,La estructura molecular del chicharo, ,02,1408,3,95
IN111696,Los virus en PC's,,02,1408,3,96
IN111797,Propiedades de los lactobacilos,1402,02,1408,3,97
cveproy,nomproy,cvedis,cvecom,cvedep,duracion,inicio
.......
```

Estos archivos en ASCII se transfirieron al servidor UNIX a través de un ftp, y una vez contando con ellos en el servidor se procedió a realizar un BCP (Bulck Copy) que no es más que un programa ejecutable que se corre desde el prompt del sistema que nos permite copiar la información en formato ASCII a las tablas de la base de datos, esto de la siguiente manera:

BCP PAPIIT : Proyecto in proyecto.txt

Se realizaron pruebas sobre la base de datos con datos reales, esto a través de los sistemas en Netscape y Power Builder, así como también a través de sentencias SQL que arrojaran los datos que se solicitaban., lo cual permitió una afinación en la base de datos.

Concluídas las pruebas procedimos a correr los scritps para el borrado del contenido de la información en las tablas y repetimos el proceso nuevamente para tener la información real con la que trabajarían os investigadores a través del sistema en Netscape.

Programa de Solicitud al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) que utiliza Netscape como front-end

Objetivo

La siguiente aplicación tiene como objetivo dar a los investigadores de la Universidad Nacional Autónoma de México, que deseen ingresar al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT), una alternativa de captura de su solicitud; usando para esto un sistema en red, en Internet, que les permita enviar su solicitud desde cualquier lugar donde se encuentren.

Propiedades de la aplicación

Las propiedades de la aplicación son las siguientes:

- Cuenta con un control de acceso al sistema (sólo pueden entrar usuarios autorizados).
- Puede ser usada desde cualquier computadora que tenga acceso a Internet y cuente con un visualizador de hipertexto (como por ejemplo, Netscape).
- El almacenamiento y recuperación de información se realiza en el manejador de bases de datos Sybase.
- El acceso a la aplicación se realiza a través de la siguiente dirección: http://tlaloc.dgapa.unam.mx/papiit/solicitud.html

Características de la aplicación

Las características de la aplicación son las siguientes:

- La captura de datos es por cada proyecto de investigación que este solicitando financiamiento de PAPIIT.
- Permite agregar, modificar y eliminar datos pertenecientes a una solicitud proyecto de investigación de PAPIIT.
- Genera reportes de la información capturada.

La aplicación está desarrollada en HTML y usa la Programación en Lenguaje C, la Programación de Clientes con Sybase y la Programación de CGI's para su desarrollo.

Desarrollo de la aplicación

1. Creación de directorios

Una vez creada la base de datos se deben de crear los directorios de trabajo para el desarrollo de la aplicación, en nuestro caso crearemos los siguientes:

/www/papiit Directorio de acceso público (podrá entrar cualquier usuario) que contendrá iconos, imágenes, documentos HTML o páginas, background's, etc.

/home/papiit/solicitud Directorio de trabajo(archivos fuente,compilación de archivos, etc.)

/home/httpd_1.4.2/cgi-bin/solicitud Directorio protegido (sólo podrán entrar usuarios autorizados) que contendrá los archivos ejecutables que serán utilizados en nuestro sistema para decodificar los documentos HTML.

 Para crear directorios sobre /www y /home/httpd_1.4.2/cgi-bin debes consultar a tu administrador de Unix.

2. Creación de prototipo

Se crearán documentos HTML o páginas (residentes en el directorio /www/papiit) que podrán ser vistos en la pantalla de la computadora como prototipo de la aplicación. Los objetivos de éste prototipo son dos: uno, que el usuario de su aprobación de la aplicación tanto en diseño como en presentación y dos, que los mismos documentos HTML nos sirvan como referencia para la programación de la aplicación (téngase que el prototipo sólo muestra el funcionamiento de la aplicación, más no hace actualizaciones, inserciones o borrado de datos).

3. Crear permisos de acceso al sistema

Debemos restringir el acceso a usuarios no autorizados, para esto debemos definir a los usuarios que van a entrar al sistema de la siguiente manera:

a) Definir el grupo de usuarios

El grupo de usuarios se define bajo el directorio /home/http/httpd_1.4.2 con un archivo que contendrá a los usuarios. Este archivo lo llamaremos *.grupo.papiit*, un fragmento de su contenido se muestra a continuación:

usuarios-papiit: IN200697
usuarios-papiit: IN300297
usuarios-papiit: IN200897
...

b) Crear contraseñas de acceso al sistema (passwords)
La utilería **htpasswd** que se encuentra bajo el directorio home/http/httpd_1.4.2/support nos permite crear las contraseñas de acceso al sistema (passwords); las contraseñas se guardarán en el archivo *.passwd.papiit* y la forma de introducirlas es la siguiente:

htpasswd [-c] /home/http/httpd_1.4.2/.passwd.papiit IN200697 JUBAM

Donde IN200697 es el usuario y "JUBAM" es la contraseña de acceso.

La opción -c se utiliza sólo en el caso de que el archivo .passwd.papiit **no** exista.


Las contraseñas de acceso tienen que ser introducidas una por una y escribirse con letras MAYÚSCULAS.

c) Definir que usuarios van a tener acceso al sistema
El archivo *.htaccess* ubicado en directorio /home/http/httpd_1.4.2/cgi-bin/papiit define los usuarios que van a tener acceso al sistema. El contenido del archivo es el siguiente:

AuthUserFile /home/http/httpd_1.4.2/**.passwd.papiit**

```
AuthGroupFile /home/http/httpd_1.4.2/.grupo.papiit
AuthName NUMERO DE PROYECTO PAPIIT
AuthType Basic

<Limit GET>
require group usuarios-papiit
</Limit>
```

Vamos a mostrar un seguimiento de la programación necesaria para poner en funcionamiento la aplicación.

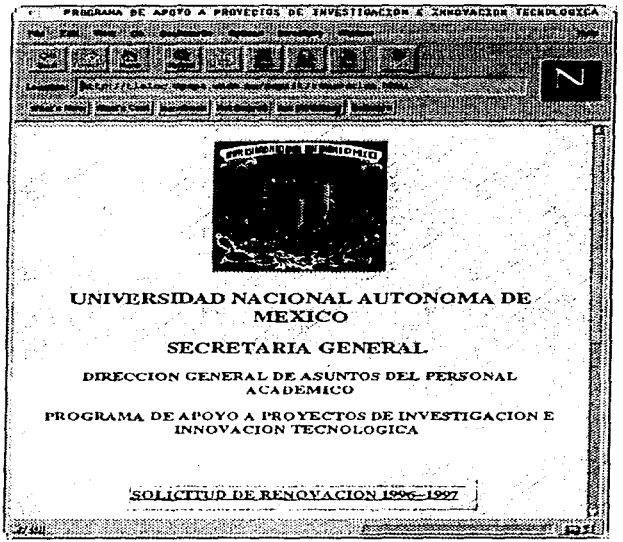
En primera instancia se coloca una página de acceso al sistema (solicitud.html) en el directorio /www/papiit (directorio de uso público) para que pueda ser accedida por cualquier usuario. Si nuestro servidor tiene el siguiente nombre tlaloc.dgapa.unam.mx entonces un usuario podrá acceder, con la ayuda de un visualizador, la página a través de la siguiente dirección: **http://tlaloc.dgapa.unam.mx/papiit/solicitud.html.**

El contenido de solicitud.html sería el siguiente:

```
<HTML>
<HEAD>
<TITLE>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
INNOVACIÓN TECNOLÓGICA</TITLE>
</HEAD>
<BODY BACKGROUND = "http://tlaloc.dgapa.unam.mx/imagenes/bk_dgapa.gif" >

<CENTER>

<IMG SRC = "escudo.gif" ALT = " imagen DE APOYO A PROYECTOS DE INVESTIGACIÓN E
INOVACIÓN TECNOLÓGICA" >
</IMG>

<H2>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</H2>
<H3>SECRETARÍA GENERAL</H3>
<H4>DIRECCIÓN GENERAL DE ASUNTOS DEL PERSONAL ACADÉMICO</H4>
<H4> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN
TECNOLÓGICA </H4>
<BLINK><H2>( PAPIIT )</H2></BLINK>
```

```
<FONT SIZE = 7 >
<PRE >
<TABLE BORDER CELLSPACING = 10 >
<TR > <TH >
<A HREF = "http://tlaloc.dgapa.unam.mx/papiit/Inicio_ren.html" >SOLICITUD DE RENOVACIÓN
1997-1998 </A >
</TH > </TR >
</TABLE >
</PRE >
</FONT >

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

Y el visualizador lo mostraría así:



Al hacer un clic sobre la liga SOLICITUD DE RENOVACIÓN 1996 - 1997

la aplicación desplegará la siguiente página:

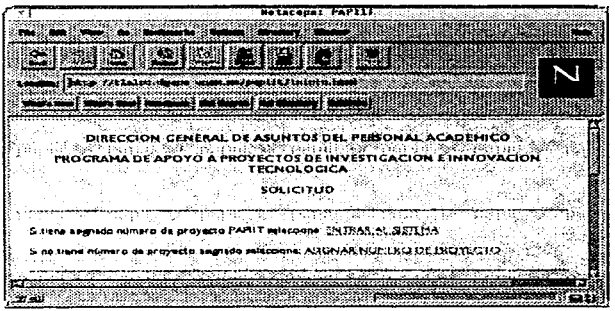
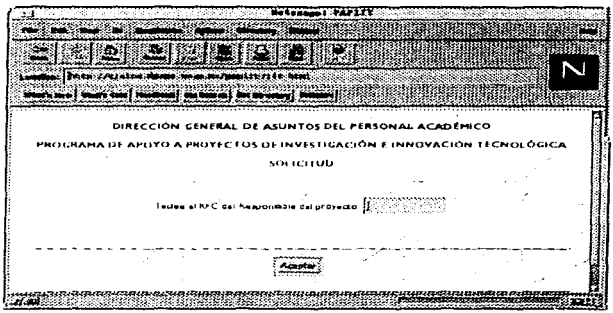


Fig. 1. Solicitud de PAPIIT

```
<HTML>
<TITLE>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
INNOVACIÓN TECNOLÓGICA
</TITLE>
<BODY BACKGROUND = "http://tlaloc.dgapa.unam.mx/imagenes/bk_dgapa.gif" >
<CENTER>
<H4>DIRECCIÓN GENERAL DE ASUNTOS DEL PERSONAL ACADÉMICO</H4>
<H4>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN
TECNOLÓGICA </H4>
<H4>SOLICITUD</H4>
<IMG SRC = "http://tlaloc.dgapa.unam.mx/iconos/lines/REDLINE1.GIF" ALING = MIDDLE>
<HR><P>
Si tiene asignado número de proyecto PAPIIT seleccione:
<A HREF = "http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/Inicio.cgi" >ENTRAR AL
SISTEMA</A><P>
Si no tiene número de proyecto asignado seleccione:
<A HREF = "http://tlaloc.dgapa.unam.mx/papiit/rfc.html" >ASIGNAR NUMERO DE
PROYECTO</A><P>
<HR>
</CENTER>
</BODY>
</HTML>
```

Si el profesor o investigador no tiene un número de proyecto asignado, tendrá que seleccionar ASIGNAR NUMERO DE PROYECTO, donde se le preguntará el RFC, como se muestra en la siguiente forma WWW (**rfc.html**):




```
<HTML>
<TITLE>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
INNOVACIÓN TECNOLÓGICA
</TITLE>
<BODY BACKGROUND = "http://tlaloc.dgapa.unam.mx/imagenes/bk_dgapa.gif" >
<CENTER>
<H4>DIRECCIÓN GENERAL DE ASUNTOS DEL PERSONAL ACADÉMICO</H4>
<H4> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN
TECNOLÓGICA </H4>
<H4>SOLICITUD</H4>
<IMG SRC = "http://tlaloc.dgapa.unam.mx/iconos/lines/REDLINE1.GIF" ALING = MIDDLE>
<HR><P>
<FORM METHOD = "POST" ACTION = "http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/rfc" >
Teclee el RFC del Responsable del proyecto: <INPUT NAME = "rfc" TYPE = "text" SIZE = 10
MAXLENGTH = 10<P>
<HR>
<INPUT TYPE = "submit" VALUE = "Aceptar" >
</CENTER>
</FORM>
</HTML>
```

En la línea:

```
<FORM METHOD = "POST" ACTION = "http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/rfc" >
```

Se manda llamar al programa CGI ejecutable llamado **rfc** (que reside en el directorio /home/httpd_1.4.2/cgi-bin/solicitud), el cual será el encargado

decodificar los datos de la forma WWW y entregar resultados de aprobación o rechazo dependiendo de lo introducido en la forma.

La estructura del programa CGI fuente **rfc** sería la siguiente:

```
/*
**
** El programa rfc.c va a decodificar la forma WWW donde se pide el RFC (rfc.html).
** A la hora de decodificar la forma va a tomar al RFC introducido por el usuario
** como entries[0].val. Esto es, RFC = entries[0].val.
**
*/
#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <string.h>

#define MAX_ENTRIES 10000
#define MAX 20000

/*
** Definición de la estructura de datos que va a guardar los valores
** capturados en una forma WWW
*/
typedef struct {
  char *name;
  char *val;
} entry;

/*
** Declaración de las funciones que decodifican una forma WWW
** (su definición está en el archivo util.c)
*/
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);


/*
** Funciones de validación de datos capturados en
** la forma WWW
*/
int cantidad(char *);              /* Validación de las cantidades solicitadas */
int cantidad(char *cant){
  int i,l;
  l=strlen(cant);
  for (i=0; i<l; i++)
    if (cant[i] < 48 || cant[i]>57)
```

```
        return 0;       /* Cantidad invalida */
    return 1;           /* cantidad valida   */
}


main(int argc, char *argv[]) {
    /*
    ** Declaración de variables
    */
    entry entries[MAX_ENTRIES];             /* Generales */
    register int x,m=0;
    int cl,dur,band;
    DBPROCESS   *dbproc,*dbproc2;
    LOGINREC    *login;
    RETCODE     return_code;
    char *nom;

    DBCHAR      rfc[10]="";

    /*
    ** Especificación de un  texto tipo HTML
    */
    printf("Content-type: text/html%c%c",10,10);

    /*
    ** Verificación del documento:
    ** debe tratarse de una forma WWW (application/x-www-form-urlencoded)
    ** y ser referenciada a trvés del metodo POST (ACTION=POST)
    ** para poder ser interpretado el documento por el programa CGI.
    */
    if (strcmp(getenv("REQUEST_METHOD"),"POST")){
       printf("Este script debe referenciarse con un metodo POST.\n");
       exit(1);
    }
    if (strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")){
       printf("Este programa solo puede ser usado para decodificar datos de una forma. \n");
       exit(1);
    }
    cl = atoi(getenv("CONTENT_LENGTH"));

   /*
   ** Decodificación de la forma:
   ** los valores se guardarán uno a uno de la siguiente manera:
   ** m tendrá el número de campos existentes en la forma
   ** entries[x].val alojará los valores de los campos capturados,
   ** entries[x].name alojará el nombre de los campos de la  forma
   ** (incluyendo los tipo HIDDEN); x=0,...,m-1
   */
   for (x=0;cl && (!feof(stdin));x++) {
      m=x;
      entries[x].val = fmakeword(stdin,'&',&cl);
      plustospace(entries[x].val);
      unescape_url(entries[x].val);
      entries[x].name = makeword(entries[x].val,'=');
```

```
    }

    /*
    ** Conexión con la base de datos Sybase
    */
    if (dbinit() == FAIL)
       exit(ERREXIT);
    login = dblogin();
    DBSETLUSER(login,"papiit");
    DBSETLPWD(login,"papiit@admin");
    dbproc = dbopen(login,NULL);
    if (dbproc == NULL){
       printf("<H1>El servidor no se encuentra arrriba</H1>");
       exit(ERREXIT);
    }
    dbuse(dbproc,"papiit");

    /*
    ** Cabecera del programa
    */

    printf("<BODY BACKGROUND=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");
    /*
    ** Verificando que se hayan capturado todos los datos pedidos
    ** en la forma WWW
    */
            if (*entries[0].val == NULL){                    /* Ver que se instroduzca el RFC
*/
       printf("<H3>Favor de teclear su RFC</H3>\n");
       exit(1);
    }

    /*
    ** Conversión a mayúsculas de los datos capturados en la forma WWW
    ** (los datos se guardarán en la base de datos con letras mayúsculas)
    */
    for(x=0; x <= m; x++){
       nom=entries[x].val;
       if(x!=8){
          while (*nom!='\0'){
             *nom=toupper(*nom);
             nom++;
          }
          while (*entries[x].val == ' '){
             entries[x].val=entries[x].val++;
             entries[x].val++;
          }
       }
    }

            /*
            ** Verificando RFC en nómina
            */
```

```
        printf("<H4>DIRECCIÓN GENERAL DE ASUNTOS DEL PERSONAL
ACADÉMICO</H4><P>\n");
  printf("<H4> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN
    TECNOLÓGICA </H4><P>\n");
  dbfcmd(dbproc,"select rfc,nombre,nomcat,nomdep
              from resp,categoria,dependencia
              where resp.cvecat = categoria.cvecat
              and resp.cvedep = dependencia.cvedep
              and rfc = '%s'",entries[0].val);
  ...
  if (dbrows(dbproc)! = SUCCEED){        /* RFC no válido */
    printf("<H3>El RFC \"%s\" no existe en la Nómina
          General</h3>\n",entries[0].val);
    printf("<H4>O bien, no tiene el nombramiento de Profesor o
          Investigador titular de tiempo completo</H4>\n");
    printf("Utilice <B>Back</B> del menú de Netscape para regresar a
          la pantalla anterior<P>\n");
    exit(ERREXIT);
  }
  ...
  strcpy(clave,"IN");
/* RFC válido (generando # de proyecto) */
  strcat(clave,entries[1].val);
  if (maximo <=9)
    strcat(clave,"00");
  if (maximo >=10 && maximo <=99)
    strcat(clave,"0");
  maximo = maximo + 2;

        /*
        ** Respuesta del programa CGI rfc en una forma WWW: despliega el número de
        ** proyecto asignado y pide el password para el proyecto. La forma será
        ** decodificada por el programa CGI rfc2. El programa rfc2 registrará el
        ** número de proyecto en la lista de proyectos como usuario autorizado
        */

  printf("<FORM METHOD=POST
        ACTION=\"http://tlaloc.dgapa.unam.mx/cgibin/solicitud/rfc2\">\n");

  printf("<H3>Número de proyecto asignado: %s\n</H3>",clave);

  printf("<INPUT TYPE=HIDDEN NAME =cveproy value=\"%s\">\n",entries[0].val);
  printf("<INPUT TYPE=HIDDEN NAME =rfc value=\"%s\">\n",rfc);
  printf("<INPUT TYPE=HIDDEN NAME =numanio value=\"%d\">\n",numanio);
  printf("Teclee su password: <INPUT TYPE=PASSWORD NAME =pass1 SIZE=8
        MAXLENGTH=8><p>\n");
  printf("Vuelva a teclearlo: <INPUT TYPE=PASSWORD NAME=pass2 SIZE=8
        MAXLENGTH=8>\n");
  printf("<BLINK><H3>Importante: Anote y conserve el Número de proyecto asignado.
        Éste y su password, serán su clave de acceso al sistema
        posteriormente.</H3></BLINK\n");
  printf("<HR>\n");
  printf("<CENTER>\n");
  printf("<INPUT TYPE=SUBMIT VALUE=\"Aceptar\">\n");
```

```
printf("</CENTER>\n");

        printf("</BODY>\n");
        printf("</FORM>\n");
        dbexit();
        exit(
}
```

Como podemos observar en la siguiente línea:

```
printf("<FORM METHOD=POST
ACTION=\"http://tlaloc.dgapa.unam.mx/cgibin/solicitud/rfc2\">\n");
```

existe un programa ejecutable (no es CGI, los programas CGI sólo decodifican formas WWW) llamado **rfc2** que será el encargado de agregar al nuevo usuario en la lista de usuarios autorizados.

Es muy importante hacer notar el paso de parámetros com objetos no visibles o "HIDDEN" (INPUT TYPE=HIDDEN), estos parámetros no son desplegados en pantalla; sin embargo, son interpretado como cualquier otro objeto de la forma. En este caso, el programa rfc2 recibirá estos parámetros cualquier otro objeto de una forma WWW, es decir, cveproy será entries[0].val, rfc será entries[1].val, etc. El uso objetos no visibles o "HIDDEN" es muy importante cuando nosotros queremos comunicar formas (pasando valores entre las formas WWW sin mostrarlos en pantalla).

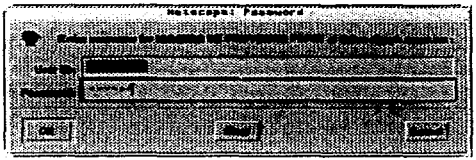
Supóngamos que profesor o investigador ya registró su proyecto y posteriormente va a capturar su solicitud.

La próxima vez que entre el profesor o investigador desea capturar su solicitud, en ese caso debe seleccionar la opción ENTRAR AL SISTEMA de la Fig. 1, notése que la liga hace referencia al siguiente archivo:

```
<A HREF="http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/Inicio.cgi">ENTRAR AL
SISTEMA</A><P>
```

El archivo en programación shell de Unix **Inicio.cgi** está en el directorio (/home/http/httpd_1.4.2/cgi-bin/solicitud) donde sólo puede entrar

usuarios autorizados, por lo tanto va a pedir autorización para entrar al sistema, como se muestra:

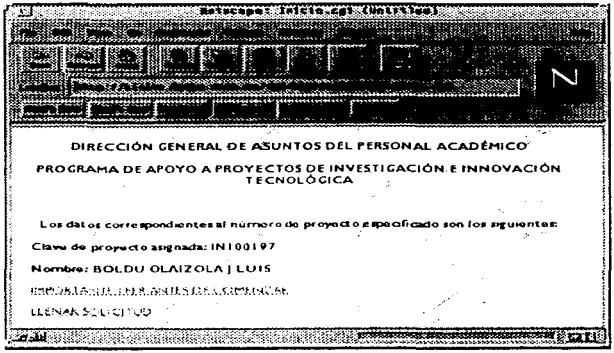


Inmediatamente despues de pedir la autorización se ejecutará el contenido del archivo **Inicio.cgi**, el cual se muestra a continuación:

```
#!/bin/sh

# Especificación de un  texto tipo HTML
echo "Content-type: text/html"
echo ""

#NOTA: El servidor HTTP manda el número de proyecto del usuario al archivo usuario_papiit
#ubicado en el directorio /tmp

#Se recupera el número de proyecto
usuario = `cat /tmp/usuario_papiit`
rm /tmp/usuario_papiit
#Se manda ejecutar el programa ejecutable rcgi para verificar que el número de proyecto
#(UserID) y la contraseña de acceso (Password) sean correctos
/home/http/httpd_1.4.2/cgi-bin/renovacion/rcgi $usuario
```

El archivo **Inicio.cgi** debe ser un archivo ejecutable, por lo que debe tener los atributo de -rwxr-xr-x, o bien, 755 (lo anterior se puede ver con el comando *ls -l Inicio.cgi* en Unix). Para cambiarle los permisos se debe de teclear el siguiente comando desde el prompt de Unix ($): *$chmod 755 Inicio.cgi*.

El objetivo de **Inicio.cgi** es recuperar el número de proyecto y verificar que sea un número válido y su password sea correcto. Sólo si el número de proyecto y el password son correctos el sistema mostrará la siguiente pantalla:




```
/*
**
** El programa rcgi es un programa cliente que verifica el número de proyecto y el password.
** argv[1] es igual al número de proyecto (recuérdese la llamada al programa rcgi $usuario,
** en donde usuario era el número de proyecto)
**
*/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <stdlib.h>
#include <time.h>


main(int argc, char *argv[]) {
  /*
  ** Declaración de variables
  */
   DBPROCESS   *dbproc;
   LOGINREC    *login;
   RETCODE     return_code;
   DBCHAR      rfc[10],numanio[2],cveproy[8];
   DBCHAR      nombre[20]="",apellidom[20]="",apellidop[20]="",titulo[12]="";

  /*
  ** Especificación de un  texto tipo HTML
  */
   printf("Content-type: text/html%c%c",10,10);

  /*
  ** Conexión con la base de datos en Sybase
  */
```

```
    if (dbinit() == FAIL)
        exit(ERREXIT);
     login = dblogin();
     DBSETLUSER(login,"papiit");
     DBSETLPWD(login,"papiit@@@");
     dbproc = dbopen(login,NULL);
     if (dbproc == NULL)
        exit(ERREXIT);

    /*
    ** Consulta a la base de datos sobre el número de proyecto
    */
    dbuse(dbproc,"papiit");
    dbfcmd(dbproc,"select p.cveproy,resp.rfc,resp.nombre,");
    dbfcmd(dbproc," resp.apellidop,resp.apellidom,resp.titulo");
    dbfcmd(dbproc," from proyecto p,responsable resp");
    dbfcmd(dbproc," where p.cveproy='%s'",argv[1]);
    dbfcmd(dbproc," and  p.cveproy=resp.cveproy");
    dbfcmd(dbproc," and resp.tiporesp='R'");
     if (dbsqlexec(dbproc) == FAIL){
       printf("Error en el servidor\n");
       dbexit();
       exit(ERREXIT);
     }
     dbresults(dbproc);
     dbbind(dbproc,1,NTBSTRINGBIND,0,cveproy);
     dbbind(dbproc,2,NTBSTRINGBIND,0,rfc);
     dbbind(dbproc,3,NTBSTRINGBIND,0,nombre);
     dbbind(dbproc,4,NTBSTRINGBIND,0,apellidop);
     dbbind(dbproc,5,NTBSTRINGBIND,0,apellidom);
     dbbind(dbproc,6,NTBSTRINGBIND,0,titulo);
     if (dbrows(dbproc)!=SUCCEED){                    /* No se encontró el proyecto */
       printf("<P>\n");
       printf("<H3>El número de proyecto no existe o el password es
incorrecto</H3><P>\n");
       printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/papiit/Inicio.html\">Verificar
datos</A>\n");
       exit(ERREXIT);
     }
     dbnextrow(dbproc);

    /*
    ** Respuesta del programa CGI rcgi
    */
    printf("<FORM>");
    printf("<BODY
            BACKGROUND=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");
    printf("<H4>DIRECCIÓN GENERAL DE ASUNTOS DEL PERSONAL
           ACADÉMICO</H4><P>\n");
    printf("<H4> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
            INNOVACIÓN TECNOLÓGICA </H4><P>\n");
    printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");
    printf("<H3>Los datos correspondientes al número de proyecto especificado son
            los siguientes:</H3>\n");
```

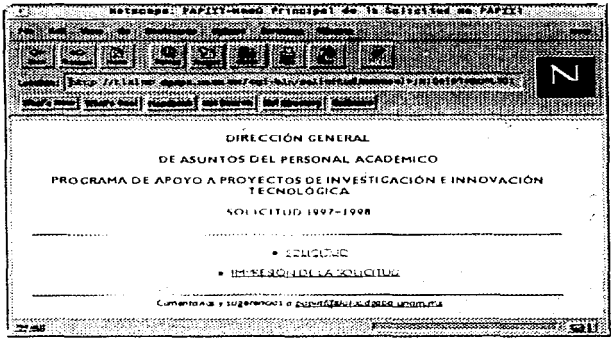
```
    printf("<H3>Clave de proyecto: %s\n</H3>",argv[1]);
    printf("<H4>RFC del responsable: %s</H4>\n",rfc);
    printf("<H4>Nombre del responsable: %s %s %s %s</H4>\n",
            titulo,nombre,apellidop,apellidom);
    printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/menusol?%s+%s+%d+
             MENUPAL\">LLENAR SOLICITUD</A><P>\n",
                ruta,argv[1],rfc,ano);
    printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/importante.cgi\">
             IMPORTANTE LEER ANTES DE COMENZAR</A>\n",ruta);
    printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");

            /*
             ** Pie del programa
             */
  printf("</BODY>\n");
  printf("</FORM>\n");
  dbexit();
  exit(ERREXIT);
}
```

Al elegir la opción de LLENAR SOLICITUD, se mandará llamar al programa ejecutable llamado **menusol** (no es un programa CGI, los programas CGI's decodifican formas WWW como resultado de oprimir un botón de *Submit* o *Aceptar*; en este caso no existe dicho botón, se hace la llamada directamente a través de la liga) con un parámtero llamado **MENUPAL**. El programa **menusol** debe residir en el directorio /home/http/httpd_1.4.2/cgi-bin/solicitud.

El objetivo de que el programa menu reciba un parámetro es por tener un solo programa que despliegue todos los menús a usar, porqué razón? por la razón de que los programas ejecutables (como **menusol**) ocupan un espacio considerado en disco (alrededor de 500 kilobytes), y si tenemos muchos programas este espacio se incrementaría considerablemente.

RECOMENDACIÓN: Las aplicaciones sólo deben de tener los mínimos programas ejecutables (en nuestra aplicación usamos tres programas).

El programa menú desplegaría algo así en pantalla:




```
/*
** El programa menusol.c crea los menús a desplegar en la apliación.  Para hacer esto
** un argumento que dependiendo de su valor realiza solo el módulo de programación
** que le corresponde. En este el caso el argumento que recibe es el argumento 4
** igual a MENUPAL (argv[4] = MENUPAL), si recibe como argumento 4 MENUSOLI
** entonces argv[4] = MENUSOLI, etc.
**
*/

#include <stdio.h>

main(int argc, char *argv[]){
   /*
   ** Especificación de un  texto tipo HTML
   */
   printf("Content-type: text/html%c%c",10,10);

   /*
   ** Cabecera de programa
   */
   printf("<BODY BACKGROUND=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");
   printf("<TITLE>PAPIIT-Menú Principal de la Renovación</TITLE>\n");
   printf("<H4>DIRECCIÓN GENERAL DE ASUNTOS DEL PERSONAL
           ACADÉMICO</H4><P>\n");
   printf("<H4> PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
            INNOVACIÓN TECNOLÓGICA </H4><P>\n");
   printf("<H3>SOLICITUD DE RENOVACIÓN 1997-1998</H3><P>\n");
   printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");
   printf("<UL>\n");
```

```
/*****************************************************************
*                          MENÚ PRINCIPAL                        *
*****************************************************************/
  if(!strcmp(argv[4],"MENUPAL")){
    printf("<LI><A HREF=\" http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud
                /menusol?%s+%s+%s+MENUSOLI\">
               SOLICITUD </A><P>\n",argv[1],argv[2],argv[3]);
    printf("<LI><A HREF=\" http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud
                /menusol?%s+%s+%s+MENUIMP\">
               IMPRESIÓN DE SOLICITUD</A><P>\n",argv[1],argv[2],argv[3]);
  }

/*****************************************************************
*                        MENÚ DE SOLICITUD                       *
*****************************************************************/
  if(!strcmp(argv[4],"MENUSOLI")){
    ...
  }

/*****************************************************************
*                      MENÚ DATOS GENERALES                      *
*****************************************************************/
  if(!strcmp(argv[4],"MENUDATGEN")){
    ...
  }

/*****************************************************************
*                         MENÚ PROTOCOLO                         *
*****************************************************************/
  if(!strcmp(argv[4],"PROTOCOLO")){
    ...
  }

/*****************************************************************
*                         MENÚ IMPRESION                         *
*****************************************************************/
  if(!strcmp(argv[4],"IMPRESION")){
    ...
  }

  ...
  /*
  ** Pie de programa
  */
  printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");
  printf("<ADDRESS>Comentarios y sugerencias a
        <A HREF=\"mailto:papiit@tlaloc.dgapa.unam.mx\">
        papiit@tlaloc.dgapa.unam.mx</A></ADDRESS>\n");
    return 0;
}
```

En la opción SOLICITUD se podrán capturar los datos referentes al proyecto de investigación que esta solicitando financiamiento del PAPIIT. Esta opción consiste los siguiente puntos:

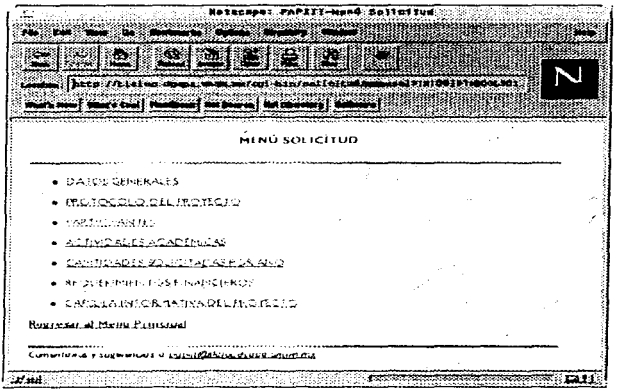


Fig.2. Menú de solicitud

El código fuente de la liga SOLICITUD (como se puede ver en menusol.c) es el siguiente:

```
printf("<LI><A HREF=\" http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud
        /menusol?%s+%s+%s+MENUSOLI\" >SOLICITUD
        </A><P>\n",argv[1],argv[2],argv[3]);
```

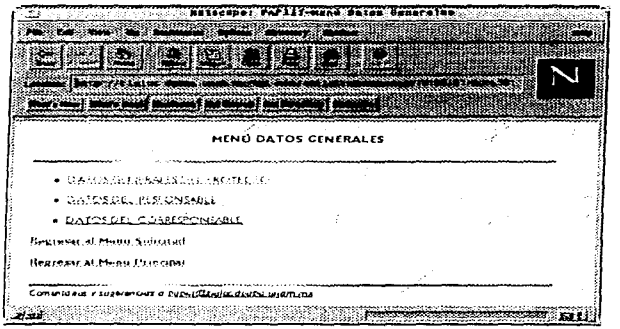
Como puede notarse, la liga mandará llamar al mismo programa **menusol** con el argumento **MENUSOLI** (el menú principal y el menú de solicitud son generados con el mismo programa menusol), el cual creará la pantalla mostrada en la fig.2.

Como puede notarse el manejo de los programas se hace con recursividad (un programa manda llamarse a si mismo), lo único que debe hacerse es llamarse con argumentos (*MENUPAL, MENUSOLI*, etc.) diferentes.

Describiremos en forma general el funcionamiento de la solicitud de PAPIIT.

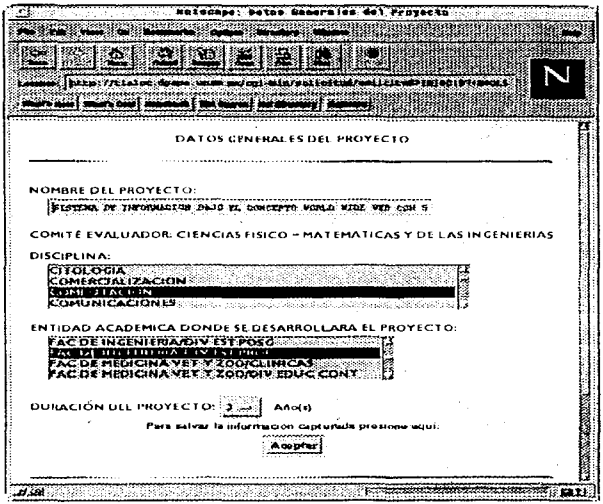
- **Datos Generales**

Los datos generales incluyen: Datos Generales del proyecto, Datos del Responsable del proyecto y Datos del Corresponsable del proyecto, como se ve en la siguiente figura:



Este menú se genera llamando al mismo programa **menusol** con el argumento **MENUDATGEN**.

Al seleccionar los DATOS GENERALES DEL PROYECTO se manda llamar al programa **solicitud** con el argumento **PROYECTO**, el resultado de esta llamada es el siguiente:




```
/*
**
** solicitud.c será el programa que creará todas las formas WWW
** utilizadas en la solicitud de PAPIIT.
** solicitud.c recibirá 4 argumentos: el argumento 1 será el número de proyecto
(argv[1] = cveproy),
** el argumento 2 será el RFC (argv[2] = rfc), el argumento 3 será el año transcurrido del
** proyecto (argv[3] = numanio) y el argumento 4 será la etiqueta que le indicará al
** programa que módulo debe realizar (argv[4] = etq).
** El programa solicitud.c en general realizará tres funciones:
** 1) Generar formas WWW de captura vacias (para agregar)
** 2) Recuperar los datos ya capturados de la base de datos y mostrarlos en una forma
**    WWW para su actualización y
** 3) Mostrar listas en donde se puedan borrar registros (participantes, convenios, etc.)
** la función que realice dependerá del argumento 4 que reciba, por ejemplo, si el argumento
4 es
** PROYECTO entonces generará la forma WWW de Datos Generales del Proyecto.
*/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <stdlib.h>
#define  MAX 20000
```

```
main(int argc, char *argv[]) {
  /*
  ** Declaración de variables
  */
  DBPROCESS   *dbproc,*dbproc2;
  DBCHAR      nomproy[250] = "",cvedis[4] = "",cvedis2[4] = "",nomdis[60] = "";

          /*
          ** Conexión a la base de datos  Sybase
          */
          if (dbinit() == FAIL)
    exit(ERREXIT);
  login = dblogin();
  DBSETLUSER(login,"papiit");
  DBSETLPWD(login,"papiit@admin");
  dbproc = dbopen(login,NULL);
  if (dbproc == NULL)
    exit(ERREXIT);
  dbproc2 = dbopen(login,NULL);
  if (dbproc2 == NULL)
    exit(ERREXIT);
  dbuse(dbproc,"papiit");

 /*
 ** Especificación de un  texto tipo HTML
 */
          printf("Content-type: text/html%c%c",10,10);

 /*
          ** Cabecera del programa
          */
  printf("<BODY BACKGROUND=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");

          /*
          ** Validando proyecto
          */
  dbfcmd(dbproc,"select * from responsable where cveproy='%s' and
            rfc='%s'",argv[1],argv[2]);
  if(dbsqlexec(dbproc) == FAIL){
    printf("Error!.\n");
    dbexit();
    exit(ERREXIT);
  }
  dbresults(dbproc);
  if(dbrows(dbproc) != SUCCEED){
    printf("<H3>NUMERO DE PROYECTO INVALIDO!.</H3>\n");
    dbexit();
    exit(1);
  }
  dbcanquery(dbproc);
```

```
/*****************************************************************
*                        DATOS GENERALES                        *
*****************************************************************/
if (!strcmp(argv[4],"PROYECTO")){
    printf("<FORM METHOD=POST
        ACTION=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/cgisol\">\n");
    printf("<TITLE>Datos Generales del Proyecto</TITLE>\n");
    printf("<CENTER><H3>DATOS GENERALES DEL PROYECTO</H3></CENTER>\n");
    printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");

    printf("<INPUT TYPE=HIDDEN NAME=cveproy
        VALUE=\"%s\">\n",argv[1]);/*IN111194 */
    printf("<INPUT TYPE=HIDDEN NAME=rfc
        VALUE=\"%s\">\n",argv[2]);/*BAMJ720624*/
    printf("<INPUT TYPE=HIDDEN NAME=numanio
        VALUE=\"%s\">\n",argv[3]);/*2*/
    printf("<INPUT TYPE=HIDDEN NAME=etiq
        VALUE=\"%s\">\n",argv[4]);/*PROYECTO*/

    /*
        ** Recuperación de datos del proyecto existentes en la base de datos
        */
    dbfcmd(dbproc,"select p.*,c.nomcom from proyecto p,comite c
            where p.cveproy='%s' and p.cvecom=c.cvecom",argv[1]);  /* argv[1]=# de
 proyecto */
    if (dbsqlexec(dbproc)==FAIL){
      printf("Error en la senetencia SQL\n");
      dbexit();
      exit(ERREXIT);
    }
    dbresults(dbproc);
    dbbind(dbproc,2,NTBSTRINGBIND,0,nomproy);
    ...
    dbbind(dbproc,5,NTBSTRINGBIND,0,nomcom);
    dbnextrow(dbproc);
    dbcanquery(dbproc);
    printf("<PRE><H4>\n");
    printf("NOMBRE DEL PROYECTO:\n");
    printf("<INPUT TYPE=TEXT NAME=nomproy SIZE=60 MAXLENGTH=250
                                        VALUE=\"%s\"><P>\n",nomproy);
    printf("COMITÉ EVALUADOR: %s<P>\n",nomcom);
    dbfcmd(dbproc,"select * from disciplina order by nomdis");
    if (dbsqlexec(dbproc) == FAIL){
      printf("Error de acceso al servidor\n");
      dbexit();
      exit(ERREXIT);
    }
    dbresults(dbproc);
    dbbind(dbproc,1,NTBSTRINGBIND,0,cvedis2);
    dbbind(dbproc,2,NTBSTRINGBIND,0,nomdis);
    printf("DISCIPLINA:\n");
    printf("<SELECT NAME=disciplina SIZE=4>\n");
    if(!strcmp(cvedis,""))              /* Si no se capturó la disciplina, no aparecerá ninguna
 seleccionada */
```

```
        while(dbnextrow(dbproc) != NO_MORE_ROWS)
          printf("<OPTION>%s\n",nomdis);
      else                                                              /*
    Si ya se capturó la disciplina, aperecerá ésta seleccionada */
        while(dbnextrow(dbproc) != NO_MORE_ROWS)
          if (!strcmp(cvedis,cvedis2))
            printf("<OPTION SELECTED>%s\n",nomdis);
          else
            printf("<OPTION>%s\n",nomdis);
      printf("</SELECT><P>\n");
      dbcanquery(dbproc);

      ...  /* Se genera el resto de la forma WWW de datos generales del proyecto */
      printf("<P><INPUT TYPE=SUBMIT VALUE=Aceptar></CENTER><P>\n");
      printf("<P><IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\"><P>\n");
      /* Comunicación entre formas WWW mediante ligas */
      printf("<B>n");
      printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
    bin/solicitud/menusol?%s+%s+%s+
          MENUDAT\">Regresar a Datos Generales del
    Proyecto</A><P>\n",argv[1],argv[2],argv[3]);
      printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
    bin/solicitud/menusol?%s+%s+%s+
          MENUSOL\">Regresar a la Solicitud</A><P>\n",argv[1],argv[2],argv[3]);
      printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
    bin/solicitud/menusol?%s+%s+%s+
          MENUPAL\">Regresar a PAPIIT</A><P>\n",argv[1],argv[2],argv[3]);
      printf("</B>\n");
          }

  /*****************************************************************
   *                          PROTOCOLO                            *
   *****************************************************************/
  if (!strcmp(argv[4],"PROTOCOLO")){
          ...
  }


  ...


  /*
   ** Pie del programa
   */
  printf("</BODY>\n");
  printf("</FORM>\n");
  dbexit();
  exit(ERREXIT);

}
```

El archivo **solicitud.c** debe pasar por los siguientes pasos para ser un programa ejecutable:

a) Incluir las siguientes líneas en el archivo makefile

```
# Demostración del uso de la herramienta make de UNIX
...
# Es muy importante separar la etiqueta (solicitud) de la instrucción $(HEADERS)
# con un tabulador. La separación con un espacio en blanco causaría un error a la
# hora de la compilación.
menusol:   $(HEADERS) menusol.c
        gcc $(INCLUDE) menusol.c $(DBLIBS) -lm -lnsl -o menusol
solicitud: $(HEADERS) soliictud.c
        gcc $(INCLUDE) solicitud.c $(DBLIBS) -lm -lnsl -o solicitud
```

b) Compilar el archivo solicitud.c con la herramienta make de Unix (para crear el programa ejecutable):

```
$make solicitud
```

c) Copiar el programa ejecutable **solicitud** al directorio /home/http/httpd_1.4.2/cgi-bin/solicitud

La forma WWW de Datos Generales del Proyecto en la siguiente línea:

```
printf("<FORM METHOD=POST
ACTION=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/cgisol\">\n");
```

indica que será **cgisol** el programa CGI que decodificará los datos capturados en ella.

Observense las líneas siguientes:

```
printf("<INPUT TYPE=HIDDEN NAME=cveproy
    VALUE=\"%s\">\n",argv[1]);
printf("<INPUT TYPE=HIDDEN NAME=rfc
    VALUE=\"%s\">\n",argv[2]);
printf("<INPUT TYPE=HIDDEN NAME=numanio
    VALUE=\"%s\">\n",argv[3]);
printf("<INPUT TYPE=HIDDEN NAME=etiq
    VALUE=\"%s\">\n",argv[4]);
```

A esto se le conoce como paso de parámetros escondidos o HIDDEN y tiene el objetivo de mantener la referencia entre formas, esto es, no perder la referencia con el registro que se está trabajando.

La línea:

```
printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
bin/solicitud/menusol?%s+%s+%s+
MENUDAT\">Regresar a Datos Generales del
Proyecto</A><P>\n",argv[1],argv[2],argv[3]);
```

sería equivalente a algo como esto:

```
printf("<A
HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
bin/solicitud/menusol?IN111194+BAMJ720624+3+
MENUDAT\">Regresar a Datos Generales del Proyecto</A><P>\n");
```

como se puede ver, esta liga le permite al usuario **regresar** al menú de Datos Generales del Proyecto *sin perder* la referencia de su proyecto que ésta capturando.

El programa **cgisol** tendría la siguiente estructura:

```
/*
**
** El programa cgisol.c es el programa encargado de decodificar las formas WWW de
** la solicitud de PAPIIT.
** El programa recibirá como entrada los objetos presentes en las formas WWW,
** incluyendo los objetos no visibles o HIDDEN, como entradas entries[#].val.
** cgisol.c recibirá siempre los siguientes 4 argumentos: el argumento 1 será el número
** de proyecto (entries[0].val=cveproy), el argumento 2 será el RFC (entries[1].val=rfc),
** el argumento 3 será el año transcurrido del proyecto (entries[2].val=numanio) y
** el argumento 4 será la etiqueta que le indicará al cual es el módulo del programa
** que se encargará de decodificar la forma WWW correspondiente (entries[3].val=etq).
** Notése que a diferencia de las formas WWW, cuyos argumentos inician desde 1 (argv[1]),
** los argumentos inician desde 0 (entries[0].val).
**
*/

#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <string.h>

#define MAX_ENTRIES 10000
#define MAX 20000

/*
** Definición de la estructura de datos que va a guardar los valores
```

```
 ** capturados en una forma WWW
 */
typedef struct {
   char *name;
   char *val;
} entry;

/*
** Declaración de las funciones que decodifican una forma WWW
** (su definición está en el archivo util.c)
*/
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);


/*
** Funciones de validación de datos capturados en
** la forma WWW
*/
int cantidad(char *);              /* Validación de las cantidades solicitadas */
int cantidad(char *cant){
   int i,l;
   l=strlen(cant);
   for (i=0; i<l; i++)
     if (cant[i] < 48 || cant[i]>57)
        return 0;      /* Cantidad invalida */
   return 1;           /* cantidad valida   */
}


main(int argc, char *argv[]) {
  /*
  ** Declaración de variables
  */
  entry entries[MAX_ENTRIES];              /* Generales */
  register int x,m=0;
  int cl,dur,band;
  DBPROCESS   *dbproc,*dbproc2;
  LOGINREC    *login;
  RETCODE     return_code;
  char *nom;

  DBCHAR      cvedis[4]="";   /* Datos generales */

  DBCHAR      antecedentes[MAX]="",objetivos[MAX];        /* Protocolo */

  /*
  ** Especificación de un  texto tipo HTML
  */
  printf("Content-type: text/html%c%c",10,10);
```

```
/*
** Verificación del documento:
** debe tratarse de una forma WWW (application/x-www-form-urlencoded)
** y ser referenciada a trvés del metodo POST (ACTION=POST)
** para poder ser interpretado el documento por el programa CGI.
*/
if (strcmp(getenv("REQUEST_METHOD"),"POST")){
   printf("Este script debe referenciarse con un metodo POST.\n");
   exit(1);
}
if (strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")){
   printf("Este programa solo puede ser usado para decodificar datos de una forma. \n");
   exit(1);
}
cl = atoi(getenv("CONTENT_LENGTH"));

/*
** Decodificación de la forma:
** los valores se guardarán uno a uno de la siguiente manera:
** m tendrá el número de campos existentes en la forma
** entries[x].val alojará los valores de los campos capturados,
** entries[x].name alojará el nombre de los campos de la  forma
** (incluyendo los tipo HIDDEN); x=0,...,m-1
*/
for (x=0;cl && (!feof(stdin));x++) {
   m=x;
   entries[x].val = fmakeword(stdin,'&',&cl);
   plustospace(entries[x].val);
   unescape_url(entries[x].val);
   entries[x].name = makeword(entries[x].val,'=');
}

/*
** Conexión con la base de datos Sybase
*/
if (dbinit() == FAIL)
   exit(ERREXIT);
login = dblogin();
DBSETLUSER(login,"papiit");
DBSETLPWD(login,"papiit@admin");
dbproc = dbopen(login,NULL);
if (dbproc == NULL){
   printf("<H1>El servidor no se encuentra arrriba</H1>");
   exit(ERREXIT);
}
dbuse(dbproc,"papiit");

dbproc2 = dbopen(login,NULL);
if (dbproc2 == NULL){
   printf("<h1>El servidor no se encuentra arrriba</h1>");
   exit(ERREXIT);
}
dbuse(dbproc2,"papiit");
```

```
/*
** Cabecera del programa
*/

printf("<body background=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");

/*
** Validando proyecto
*/
dbfcmd(dbproc,"select * from responsable1 where cveproy='%s' and
            rfc='%s'",entries[0].val,entries[1].val);
if (dbsqlexec(dbproc) == FAIL){
   printf("Error!.\n");
   dbexit();
   exit(ERREXIT);
}
dbresults(dbproc);
if (dbrows(dbproc) != SUCCEED){
   printf("<H3>NUMERO DE PROYECTO INVALIDO!.</H3>\n");
   dbexit();
   exit(1);
}

/*****************************************************************
*                        DATOS GENERALES DEL PROYECTO             *
*****************************************************************/
if (!strcmp(entries[3].val,"PROYECTO")){
    printf("<TITLE>Datos Generales del Proyecto</TITLE>\n");
    printf("<CENTER><H3>DATOS GENERALES DEL PROYECTO</H3></CENTER>\n");
    printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");

    /*
    ** Verificando que se hayan capturado todos los datos pedidos
    ** en la forma WWW
    */
    if (*entries[5].val == NULL){
       printf("<h3>Favor de escribir el nombre del proyecto</h3><P>\n");
       printf("Utilice <B>\"Back\"</B> del menú de Netscape para regresar a la pantalla
                                        anterior<P>\n");
       exit(1);
    }
    if (m != 8){                /* Son 8 los campos los que se deben capturar en la forma
  WWW */
       printf("<H3>Favor de seleccionar la disciplina y Entidad Académica donde se está
                                        desarrollando      el proyecto</H3><P>\n");
       printf("Utilice <B>\"Back\"</B> del menú de Netscape para regresar a la pantalla
                                        anterior<P>\n");
       exit(1);
    }


    /*
    ** Conversión a mayúsculas de los datos capturados en la forma WWW
    ** (los datos se guardarán en la base de datos con letras mayúsculas)
```

```
    */
    for(x=0; x <= m; x++){
      nom=entries[x].val;
      if(x!=8){
        while (*nom!='\0'){
          *nom=toupper(*nom);
          nom++;
        }
        while (*entries[x].val == ' '){
          entries[x].val=entries[x].val++;
          entries[x].val++;
        }
      }
    }

    /*
    ** Actualización de la base de datos
    */
    dbcanquery(dbproc);
    dbfcmd (dbproc,"select cvedis,cvedep from disciplina,dependencia
                where nomdis=\"%s\" and nomdep=\"%s\"",
                entries[6].val,entries[7].val);                    /* Obteniendo claves de
disciplina y dependencia */
    if (dbsqlexec(dbproc) == FAIL){
      printf("<H2>Error durante el proceso de acceso al servidor</H2>");
      dbexit();
      exit(ERREXIT);
    }
    dbresults(dbproc);
    dbbind(dbproc,1,NTBSTRINGBIND,0,cvedis);
    dbbind(dbproc,2,NTBSTRINGBIND,0,cvedep);
    if (dbrows(dbproc) != SUCCEED){                /* Verificando contenido de catálogos */
      printf("<P>");
      printf("<H2>No hay información en los catálogos de Disciplina y
Dependencias</H2> ");
      printf("<P>");
      exit(1);
    }
    dbnextrow(dbproc);
    dbcanquery(dbproc);                                          /* Actualizando (update) */
    dbfcmd(dbproc,"update proyecto set nomproy='%s',cvedis='%s',cvedep='%s',
              duracion=%s where cveproy='%s'",
              entries[5].val,cvedis,cvedep,entries[8].val,entries[0].val);
    if (dbsqlexec(dbproc) == FAIL){
      printf("<H2>Error durante el proceso de acceso al servidor</H2>");
      dbexit();
      exit(ERREXIT);
    }


  /*
  ** Respuesta del programa CGI a través de una forma WWW
  */
```

```
    printf("<H3>Los datos generales del proyecto son los siguientes:</H3>\n");
    printf("<PRE><H4>\n");
    printf("<P><P>\n");
    printf("Clave del Proyecto: <B>%s</B><P>\n",entries[0].val);
    printf("Nombre del Proyecto: <B>%s</B><P> \n", entries[5].val);
    printf("Comité evaluador: <B>%s</B><P>\n",entries[4].val);
    printf("Disciplina: <B>%s</B><P>\n",entries[6].val);
    printf("Entidad Académica donde se desarrollará el proyecto:
  <B>%s</B><P>\n",entries[7].val);
    printf("Duración del proyecto: <B>%s años(s)</B><P>\n",entries[8].val);
    printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");
    printf("<B><P>\n");
    printf("<B>n");                    /* Comunicación entre formas WWW mediante ligas
  */
    printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
  bin/solicitud/menusol?%s+%s+%s+
        MENUDAT\">Regresar a Datos Generales del
        Proyecto</A><P>\n",entries[0].val,entries[1].val,entries[2].val);
}

/*****************************************************************
*                          PROTOCOLO                             *
*****************************************************************/
if (!strcmp(entries[3].val,"PROTOCOLO")){

        ...

}


        ...


/*
 ** Pie del programa
 */
printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/menusol?%s+%s+%s+
        MENUSOL\">Regresar a la
        Solicitud</A><P>\n",entries[0].val,entries[1].val,entries[2].val);
printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/menusol?%s+%s+%s+
        MENUPAL\">Regresar a
        PAPIIT</A><P>\n",entries[0].val,entries[1].val,entries[2].val);
printf("</B>\n");
printf("</BODY>\n");
printf("</FORM>\n");
dbexit();
exit(ERREXIT);
}
```

El archivo **cgisol.c** debe pasar por los siguientes pasos para ser un programama ejecutable:

a) Incluir las siguientes líneas en el archivo makefile (observése que se debe incluir el archivo objeto del archivo util.c, pués como se vió util.c contiene las funciones que van a decodicar las formas WWW)

```
# Demostración del uso de la herramienta make de UNIX
...
# Es muy importante separar las etiquetas (solicitud, etc.) de la instrucción $(HEADERS)
# con un tabulador. La separación con un espacio en blanco causaría un error a la
# hora de la compilación.
menusol:   $(HEADERS) menusol.c
        gcc $(INCLUDE) menusol.c $(DBLIBS) -lm -lnsl -o menusol
cgisol: $(HEADERS) cgisol.c
        gcc $(INCLUDE) cgisol.c util.o $(DBLIBS) -lm -lnsl -o cgisol
```

b) Compilar el archivo solicitud.c con la herramienta make de Unix (para crear el programa ejecutable solicitud):

```
$make cgisol
```

c) Copiar el programa ejecutable **cgisol** al directorio /home/http/httpd_1.4.2/cgi-bin/solicitud

Estas líneas:

```
printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/menusol?%s+%s+%s+
        MENUDAT\">Regresar a Datos Generales del
        Proyecto</A><P>\n",entries[0].val,entries[1].val,entries[2].val);
                ...
        /*
        ** Pie del programa
        */
        printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
bin/solicitud/menusol?%s+%s+%s+
                MENUSOL\">Regresar a la
                Solicitud</A><P>\n",entries[0].val,entries[1].val,entries[2].val);
        printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
bin/solicitud/menusol?%s+%s+%s+
                MENUPAL\">Regresar a
                PAPIIT</A><P>\n",entries[0].val,entries[1].val,entries[2].val);
```

le permite al usuario **regresar** a un menú anterior (sin perder la referencia del proyecto) para seguir capturando datos.

La **respuesta del programa CGI a través de una forma WWW** se mostraría en una forma HTML en Netscape.

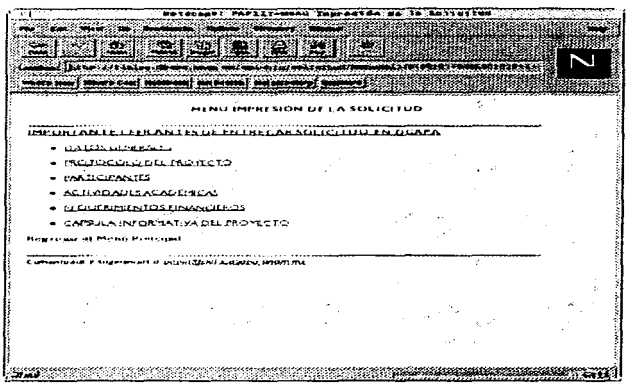
Como se puedó ver la aplicación utiliza tres programas ejecutables, residentes en el directorio /home/http/httpd_1.4.2/cgi-bin/solicitud, que resumiremos a continuación:

- **menu:** Crea los menús a utilizar en la aplicación.

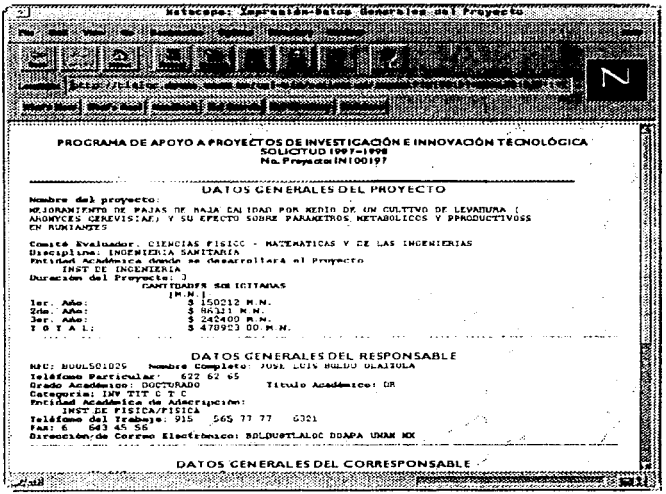
- **solicitud:** Crea las formas WWW de la solicitud PAPIIT.
- **cgisol:** Decodifica las formas WWW de la solicitud PAPIIT y actualiza, agrega o borra los datos existentes en la base de datos Sybase.
- **impsol:** Crea los reportes de los datos capturados en el proyecto para impresión (este programa lo veremos más adelante).

Los demás rubros de la aplicación tienen el mismo procedimiento del rubro de Datos Generales del Proyecto: primero, se crea el menú de opciones en el archivo **menu.c**; segundo, se crea la forma WWW del rubro en el archivo **solicitud.c** y, tercero, se crea el programa CGI que decodifica y da una respuesta en el archivo **cgisol.c**. Cada que exista una modificación a cualquiera de los tres archivos mencionados anteriormente se tiene que compilar el programa y copiarlo al directorio especificado anteriormente.

En el menú principal al elegir IMPRESIÓN DE LA SOLICITUD se mandaría ejecutar el programa menú con el parámetro "IMPRESION", generando una pantalla como la que sigue:



Si se eligen los DATOS GENERALES, la aplicación generaría la siguiente salida en pantalla lista para ser impresa:



PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E INNOVACIÓN TECNOLÓGICA
SOLICITUD 1997-1998
No. Proyecto IN100197

DATOS GENERALES DEL PROYECTO

Nombre del proyecto:
MEJORAMIENTO DE PAJAS DE BAJA CALIDAD POR MEDIO DE UN CULTIVO DE LEVADURA (
ANDMYCES CEREVISIAE) Y SU EFECTO SOBRE PARAMETROS METABOLICOS Y PRODUCTIVOSS
EN RUMIANTES

Comité Evaluador: CIENCIAS FISICO - MATEMATICAS Y DE LAS INGENIERIAS
Disciplina: INGENIERIA SANITARIA
Entidad Académica donde se desarrollará el Proyecto
INST DE INGENIERIA
Duración del Proyecto: 3
CANTIDADES SOLICITADAS
(M.N.)
1er. Año: $ 150212 M.N.
2do. Año: $ 86331 M.N.
3er. Año: $ 242400 M.N.
T O T A L: $ 478943.00 M.N.

DATOS GENERALES DEL RESPONSABLE

RFC: BOOL501029 Nombre Completo: JOSE LUIS BOLDO OLAIZOLA
Teléfono Particular: 622 62 65
Grado Académico: DOCTORADO Título Académico: DR
Categoría: INV TIT C T C
Entidad Académica de Adscripción:
INST DE FISICA/FISICA
Teléfono del Trabajo: 915 565 77 77 6321
Fax: 6 563 45 56
Dirección de Correo Electrónico: BOLDO@TLALOC.DGAPA.UNAM.MX

DATOS GENERALES DEL CORRESPONSABLE

```
/*
** impsol.c es el programa cliente que va a generar reportes para impresión
** del proyecto.
*/
#include <stdio.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <time.h>
#include <string.h>
#include <time.h>

#define MAX 200000
#define max 80
#define  REQ 9
DBCHAR rfc[10];

/*
** Función que divide un parrafo en renglones
*/
void divide(char *);
```

```
void divide(char parrafo[]){
  int i,j,k,l,m,n,p;
  l = strlen(parrafo);
  if (parrafo[max] = = ' '){
    for (m = 0;m < = max;m + +)
      printf("%c",parrafo[m]);
    printf("\n");
    if (l > max * 2)
    if (parrafo[max * 2] = = ' '){
      for (m = max + 1;m < = max * 2;m + +)
        printf("%c",parrafo[m]);
      printf("\n");
      for (m = (max * 2 + 1);m < = l;m + +)
        printf("%c",parrafo[m]);
      printf("\n");
    }
    else{
      m = max * 2;
      while(parrafo[m]! = ' ')
        m--;
      for (j = max + 1;j < = m;j + +)
        printf("%c",parrafo[j]);
      printf("\n");
      for (j = m + 1;j < = l;j + +)
        printf("%c",parrafo[j]);
      printf("\n");
      }
      else {
        for (m = max + 1;m < = l;m + +)
          printf("%c",parrafo[m]);
        printf("\n");
      }
    }
  else{
    m = max;
    while(parrafo[m]! = ' ')
      m--;
    for (j = 0;j < = m;j + +)
      printf("%c",parrafo[j]);
    printf("\n");
    if (l > max * 2)
      if (parrafo[max * 2] = = ' '){
        for (j = m + 1;j < = max * 2;j + +)
          printf("%c",parrafo[j]);
        printf("\n");
        for (j = (max * 2 + 1);j < = l;j + +)
          printf("%c",parrafo[j]);
        printf("\n");
      }
      else{
        m = max * 2;
        while(parrafo[m]! = ' ')
          m--;
        for (j = max + 1;j < = m;j + +)
```

```
            printf("%c",parrafo[j]);
          printf("\n");
          for (j=m+1;j<=l;j++)
            printf("%c",parrafo[j]);
          printf("\n");
        }
      else {
        for (j=m+1;j<=l;j++)
          printf("%c",parrafo[j]);
        printf("\n");
      }
    }
}

main(int argc, char *argv[]){
          /*
          ** Declaración de variables
          */
  DBPROCESS *dbproc,*dbproc2;
  LOGINREC  *login;
  RETCODE   return_code;

          DBCHAR    nomproy[250]="",nomdis[61]="",nomcom[51]="";        /*    Datos
Generales del Proyecto, */
  DBCHAR    nomdep[41],duracion[2]="",numanio[2]="";                    /*
Responsable y Corresponsable */
  DBCHAR    cantsol[10]="",nombre[25]="",apellidop[20]="";
  DBCHAR    apellidom[20]="",ladapart[7]="",telpart[9]="";
  DBCHAR    grado[12]="",titulo[11]="",ladatrab[7]="";
  DBCHAR    teltrab[9]="",extension[5]="",ladafax[7]="";
  DBCHAR    numfax[9]="",dircorreo[60]="",nomcat[15]="";
  DBCHAR    director[60]="";

          /*
          ** Conexión a la base de datos Sybase
          */
          if (dbinit() == FAIL)
    exit(ERREXIT);
  login = dblogin();

  DBSETLUSER(login,"papiit");
  DBSETLPWD(login,"papiit@admin");
  dbproc = dbopen(login,NULL);
  if (dbproc == NULL)
    exit(ERREXIT);
  dbuse(dbproc,"papiit");

  /*
  ** Especificación de un  texto tipo HTML
  */
  printf("Content-type: text/html%c%c",10,10);

          /*
          ** Cabecera del programa
```

```
            */
            printf("<body
 background=\"http://tlaloc.dgapa.unam.mx/papiit/rbackmenu.gif\">\n");

            /*
            ** Validando proyecto
            */
            dbfcmd(dbproc,"select * from responsable1 where cveproy='%s'",argv[1]);
      dbfcmd(dbproc," and rfc='%s'",argv[2]);
      if(dbsqlexec(dbproc) == FAIL){
        printf("Error!.\n");
        dbexit();
        exit(ERREXIT);
      }
      dbresults(dbproc);
      if(dbrows(dbproc) != SUCCEED){
        printf("<h3>NUMERO DE PROYECTO INVALIDO!.</h3>\n");
        dbexit();
        exit(1);
      }
      dbcanquery(dbproc);

   /*****************************************************************
    *          DATOS GENERALES DEL PROYECTO, RESPONSABLE Y CORRESPONSABLE      *
    *****************************************************************/
   if(!strcmp(argv[4],"DAT_GEN")){
      printf("<TITLE>Impresión - Datos Generales del Proyecto</TITLE>\n");
      printf("<PRE>\n");
      printf("<CENTER><H5>PROGRAMA DE APOYO A PROYECTOS DE INVESTIGACIÓN E
                                  INNOVACIÓN TECNOLÓGICA\n");
                        printf("SOLICITUD 1997-1998\n",punt->tm_year+1,punt->tm_year+2);
               printf("No. Proyecto: %s </H5></CENTER>\n",argv[1]);
            printf("<IMG SRC=\"http://tlaloc.dgapa.unam.mx/papiit/linemenu.gif\">\n");
              dbcanquery(dbproc);
            dbfcmd(dbproc,"select p.*,d.nomdis,c.nomcom,de.nomdep
               from proyecto p,disciplina d,comite c,dependencia de
               where p.cveproy='%s' and p.cvecom=c.cvecom
               and p.cvedis=d.cvedis and p.cvedep=de.cvedep",
               argv[1]);
            if (dbsqlexec(dbproc) == FAIL){
                        printf("Error de acceso al servidor\n");
      dbexit();
            }
            dbresults(dbproc);
            dbbind(dbproc,2 ,NTBSTRINGBIND,0,nomproy);
            dbbind(dbproc,6 ,NTBSTRINGBIND,0,duracion);
            dbbind(dbproc,8 ,NTBSTRINGBIND,0,nomdis);
            dbbind(dbproc,9 ,NTBSTRINGBIND,0,nomcom);
            dbbind(dbproc,10,NTBSTRINGBIND,0,nomdep);
            if (dbrows(dbproc)!=SUCCEED){/* No hay información disponible */
                        printf("<h3>No hay información en Datos generales del
proyecto</h3>\n");
                        dbexit();
                        exit(ERREXIT);
```

```
        }
        dbnextrow(dbproc);


                /*
                ** DATOS GENERALES DEL PROYECTO
                */
        printf("<H4><CENTER>DATOS                GENERALES                DEL
PROYECTO</CENTER></H4><P>");
        printf("<B>Nombre del proyecto:</B>\n");/* Puesto que el nombre del proyecto
puede */
        divide(nomproy);
        /* ser de varios renglones, se llama a una función */
        /* para que divida el nombre en varios renglones */
          printf("<P>\n");
                printf("<B>Comité Evaluador:</B>  %s \n",nomcom);
        printf("<B>Disciplina:</B>  %s \n",nomdis);
        printf("<B>Entidad Académica donde se desarrollará el Proyecto</B>\n");
          printf("     %s<P>\n",nomdep);
        printf("<B>Duración del Proyecto:</B>  %s\n",duracion);

                /*
                ** CANTIDADES SOLICITADAS
                */
                ...

                /*
                ** DATOS GENERALES DEL RESPONSABLE
                */
                ...

                /*
                ** DATOS GENERALES DEL CORRESPONSABLE
                */
                ...
        }

 /*************************************************************************
 *                                PROTOCOLO                              *
 *************************************************************************/
 if(!strcmp(argv[4],"PROTOCOLO")){
        ...
        }

        /*
        ** Pie del programa
        */
 printf("<B>n");
  printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-bin/solicitud/menusol?%s+%s+%s+
        MENUDAT\">Regresar al menú de impresión
</A><P>\n",argv[1],argv[2],argv[3]);
        printf("<A HREF=\"http://tlaloc.dgapa.unam.mx/cgi-
bin/solicitud/menusol?%s+%s+%s+
        MENUPAL\">Regresar a PAPIIT</A><P>\n",argv[1],argv[2],argv[3]);
```

```
  printf("</B>\n");
  printf("</BODY>\n");
  printf("</FORM>\n");
  dbexit();
  exit(ERREXIT);
}
```

El archivo **impsol.c** debe pasar por los siguientes pasos para ser un programa ejecutable:

a) Incluir las siguientes líneas en el archivo makefile

```
# Demostración del uso de la herramienta make de UNIX
...
# Es muy importante separar la etiquetas (impsol) de la instrucción $(HEADERS)
# con un tabulador. La separación con un espacio en blanco causaría un error a la
# hora de la compilación.
menusol:   $(HEADERS) menusol.c
        gcc $(INCLUDE) menusol.c $(DBLIBS) -lm -lnsl -o menusol
impsol: $(HEADERS) impsol.c
        gcc $(INCLUDE) impsol.c  $(DBLIBS) -lm -lnsl -o impsol
```

b) Compilar el archivo impsol.c con la herramienta make de Unix (para crear el programa ejecutable solicitud):

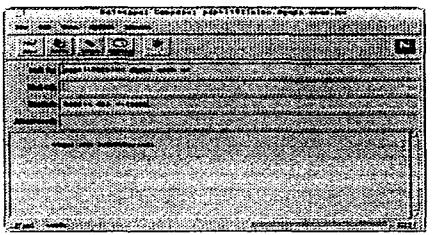
```
$make impsol
```

c) Copiar el programa ejecutable **impsol** al directorio /home/http/httpd_1.4.2/cgi-bin/solicitud

La opción del correo electrónico, se incluye para captar dudas, observaciones o fallas, referentes al sistemas; ésta se vería así en el sistema:

Comentarios y sugerencias a papiit@tlaloc.dgapa.unam.mx

Y este sería el resultado de habilitarla (con un clic sobre la liga papiit@tlaloc.dgapa.unam.mx):



❖ Programa de Consulta por Comités del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) que utiliza PowerBuilder como front-end

Objetivo

La siguiente aplicación GUI tiene como objetivo generar reportes y gráficas por comité de los datos capturados en el la solicitud de ingreso al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) a través de Internet. Como antecedente, recordemos que los proyectos que participan en el PAPIIT están ubicados en uno de cinco comités existentes.

La aplicación está desarrollada con una herramienta de desarrollo gráfico conocida como PowerBuilder Enterprise 5.0.

Propiedades de la aplicación

Las propiedades de la aplicación son las siguientes:

- Es una aplicación gráfica (GUI).
- Accesa datos de un manejador de base de datos conocido como Sybase.
- Está diseñada para ejecutarse en una máquina que tenga Windows 3.x o Windows 95.

Características de la aplicación

Las características de la aplicación son las siguientes:

- Genera reportes
- Genera gráficas y
- Exporta datos (de Sybase a dBase o Excel)

Requerimientos de la aplicación

Los requerimientos de la aplicación son tener:

- Windows 3.x o Windows 95
- PowerBuilder Enterprise 5.0
- Open Client for Windows (sólo si se hace acceso a Sybase)

Para más información de Open Client for Windows consulte a su distribuidor de Sybase.

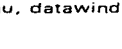
En nuestra aplicación, PowerBuilder utilizará la base de datos Sybase; por tanto, es indispensable tener instalado Open Client for Windows y tener definidas las siguientes variables de ambiente en nuestro en nuestra computadora:

```
LANG = enu
SYBASE = C:\SQL10
DSQUERY = SYBASE
PATH = C:\SQL10\BIN;C:\SQL10\DLL
LIB = C:\SQL10\LIB
INCLUDE = C:\SQL10\INCLUDE
```

Si no se tienen definidas, véase el tema "Programación con PowerBuilder -> Variables de ambiente" del capítulo 5 de este libro.

Desarrollo de la aplicación

Para el desarrollo de la aplicación es importante leer el tema de "Programación con PowerBuilder" del capítulo 5 de este libro.

Las propiedades (color, tipo de letra, icono, cursor, etc.) de los objetos (application, window, menu, datawindow, etc.) se especifican dando un clic sobre el painter de propiedades (), o bien, dando un clic derecho sobre el y después elegir *Propierties...* ().

PowerBuilder usa el lenguaje de programación *PowerScript*, por lo que al escribir un script ( ) **se debe compilar**, para ver si no tiene errores de programación. La compilación con el painter de "Return" (es el último painter de la barra ubicada a la derecha), la apariencia del painter "Return" depende del script que se este escribiendo, si es el script de una aplicación este se vería así ( ), si es el de un objeto de una ventana se vería así ( ), etc.

Los objetos guardan desde la opción Save del menú File (File -> Save).

Paso 1

Creamos un una aplicación nueva, llamada *comite.pbl*, usando el painter Application ( ). La aplicación no debe tener template, esto es, que cuando nos pregunte:

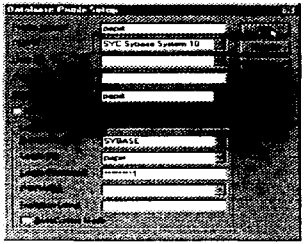
Elijamos "No".

Elegimos el icono para la aplicación desde la opción "Icon" de las propiedades del objeto, en el ejemplo el icono elegido fue el siguiente: .

Paso 2

Verificar la conexión con el servidor de bases de datos Sybase (o bien, cualquier otro), usando el painter Database ( ).

Seleccionamos del menú "File -> Connect... -> Setup..." el servidor de base de datos y la base de datos.



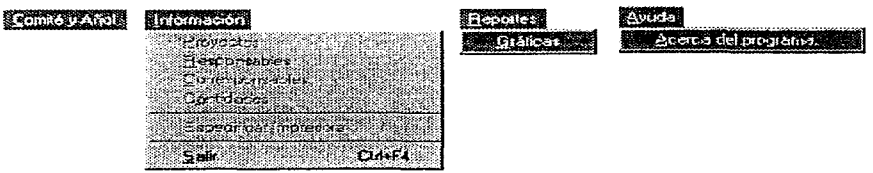
En este momento ya se estableció la conexión con el servidor de bases de datos y permanecerá así hasta que elijamos otra fuente de datos.

La conexión con la base de datos nos servirá para crear interfaces con los datos (*datawindows*).

Paso 3

Creación de menús usando el painter Menu ().

Creamos el menú principal con las siguientes opciones:



Observensen las opciones que están deshabilitadas (el atributo de *enabled* no debe estar habilitado).

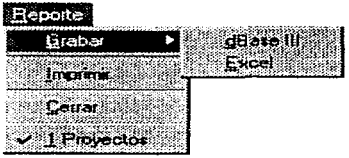
Guardamos el menú con el nombre de **m_principal.**

En el script de Salir escribimos lo siguiente:

```
// Cierra todas las conexiones existentes
// a la base de datos y sale del sistema
HALT CLOSE
```

Creamos un segundo menú con las siguientes opciones:

Y lo guardamos con el nombre de **m_export**.



 Paso 3

Creamos nuestra ventana principal usando el painter Window (). La ventana debe ser tipo "MDI Frame with microhelp". A ésta ventana le asociamos el menú **m_principal** (recuérdese que un menú se asocia a una ventana desde las propiedades de ésta, dando doble clic a la ventana podemos ver las propiedades).

Guardamos la ventana con el nombre de **w_principal**.

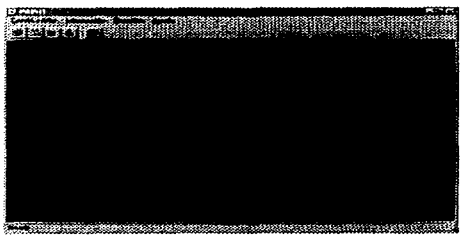
 ***Prueba de la aplicación***

Vamos a comunicar los objetos que hemos realizado para correr nuestra aplicación y ver si es así como queremos que se vea.

Al objeto de *application* () le ponemos el siguiente *script*:

```
/**********************************************************
** Script para la aplicación de consultas por comités de PAPIIT
** Lenguaje de Programación: PowerScript
** Autores: jbm, yam, acdm
** Fecha: Enero de 1997
**********************************************************/
// Atributos del objeto de transacción SQLCA
sqlca.DBMS              = "SYC Sybase System 10"
sqlca.database                  = "papiit"
sqlca.userid            = "papiit"
sqlca.dbpass            = "papiit@admin"
sqlca.logid             = "papiit"
sqlca.logpass                   = "papiit@admin"
sqlca.servername        = "SYBASE"
// Estableciendo una conexión a la base de datos
connect;
// Verificación de la conexión
if sqlca.sqlcode < > 0 then
   MessageBox ("No se pudo realizar la conexión ", sqlca.sqlerrtext)
   return
end if
// Desconexión de la base de datos
disconnect;
// Variables globales
year = 97
numanio = 1
comite = '1'
// Llamada a la ventana principal
Open (w_principal)
```

Al oprimir el painter Run () veríamos una pantalla como está:

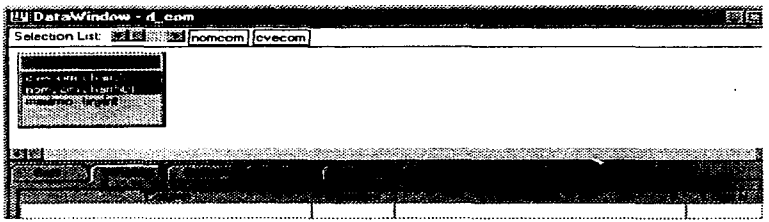


Oprimimos el painter Salir () para regresar al diseño de la aplicación en PowerBuilder.

Las opciones que aparecen deshabilitadas permanecerán así hasta que elijamos el "Año y Comité". Veremos a continuación como podemos crear está opción.

Paso 4

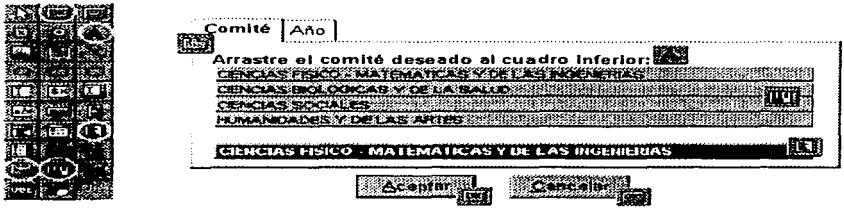
Creamos un ***datawindow sin argumentos*** **SQL Select** tipo **Free Form** con los campos siguientes: la clave del comité (*cvecom*) y el nombre del comité (*nomcom*) de la **tabla comite**.



Guardamos el datawindow con el nombre de **d_com**.

Paso 5

Creamos una ventana tipo "Response" que tenga la siguiente apariencia:



El datawindow pegado es el previamente hecho **d_com**.

El script en el *evento dragdrop* para la lista desplegable ( ) sería el siguiente:

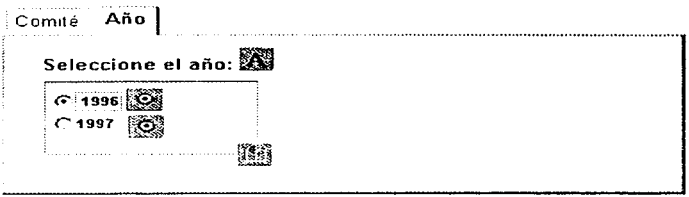
```
// Inicia arrastre de objetos
This.Drag(Begin!)
// Sólo permite un nombre en la lista
if lb_1.TotalItems() = 1 Then
   lb_1.deleteItem(1)
   lb_1.AddItem(cve)
else
   lb_1.AddItem(cve)
end if
// Asigna comité
IF This.text(1) = 'CIENCIAS FISICO - MATEMATICAS Y DE LAS INGENIERIAS' THEN
   comitebak = "1"
ELSEIF This.text(1) = 'CIENCIAS BIOLOGICAS Y DE LA SALUD' THEN
   comitebak = "2"
ELSEIF This.text(1) = 'CIENCIAS SOCIALES' THEN
   comitebak = "3"
ELSEIF This.text(1) = 'HUMANIDADES Y DE LAS ARTES' THEN
   comitebak = "4"
ELSE
   comitebak = "5"
END IF
// Deshabilita arrastre de objetos
This.Drag(End!)
```

Para el datawindow ( ) los scripts serían, para el *evento rowfocuschanged*:

```
// Guardar el nombre del comité seleccionado
cve = This.getitemstring(This.getrow(),1)
```

y para el *evento clicked*:

```
// Iniciar arrastre de objetos
This.Drag(Begin!)
```



El script para el radioboton 1996 sería el siguiente:

```
// Fija el año si está seleccionado
IF checked THEN
   year = 96
END IF
```

Para el radioboton sería algo parecido, sólo que year = 97.

Para el botón de "Aceptar", el script en el *evento clicked* sería el siguiente:

```
 // Activa el cursor de reloj (espera)
setpointer(hourglass!)
// Si cambia de comite cierra las ventanas abiertas
IF comite <> comitebak THEN
   close(w_proy)
   close(w_resp)
   close(w_corresp)
   close(w_cantidad)
END IF
// Asigna el comité
comite = comitebak
// Habilita las opciones del menú
m_principal.m_informacin.enabled = true
m_principal.m_informacin.m_a.enabled = true
m_principal.m_informacin.m_b.enabled = true
m_principal.m_informacin.m_c.enabled = true
m_principal.m_informacin.m_d.enabled = true
m_principal.m_informacin.m_e.enabled = true
// Cierra la ventana
close(parent)
```

Para el botón de cancelar el script en el *evento clicked* sería este:

```
// Cierra la ventana
close(parent)
```

Finalmente, y muy importante, como la ventana utiliza un datawindow (recupera información de la base de datos) se requiere el siguiente script en el *evento constructor* del objeto tab ():

```
// NOTA: dw_1 es el nombre del objeto datawindow dentro de
// una ventana (este nombre puede ser cambiado, por ejemplo a dw_comite)
// Establece una conexión con el servidor
connect;
// Define el objeto de transacción a usar
dw_1.settransobject(SQLCA)
// Recupera la información
dw_1.retrieve()
// Cierra la conexión
disconnect;
```

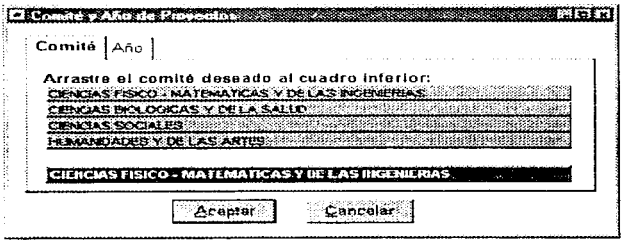
Como se ve, una vez que se haya elegido un Comité y un Año se habilitarán todas las opciones del menú.

Guardamos la ventana con el nombre de w_comite y colocamos el siguiente script en la opción de "Comité y Año" del menú principal (m_principal):

```
// Activa el cursor de reloj (espera)
setpointer(hourglass!)
// Abrimos la ventana de comité y año
open(w_comite)
```

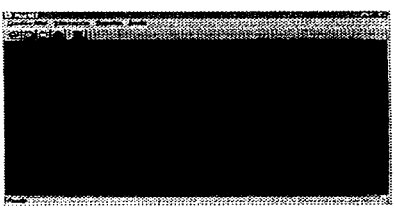
Prueba de la aplicación

Al oprimir el painter Run (), elegimos la opción "Comité y Año" del menú principal:



En está ventana el usuario tiene que seleccionar el comité y el año corresponde al año en que se registraron los proyectos.

Una vez que el usuario ha elegido el comité y año se habilitan las opciones antes deshabilitadas (obsérvese que la barra de herramientas - ubicada justamente abajo de la barra de menúes- aparece habilitada en comparación con la pantalla principal mostrada al inicio) como se muestra a continuación.

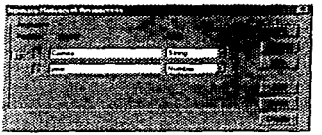


En la opción de Información el usuario puede seleccionar uno de los siguientes reportes, así como especificar la impresora que va a utilizar para imprimir los mismos.

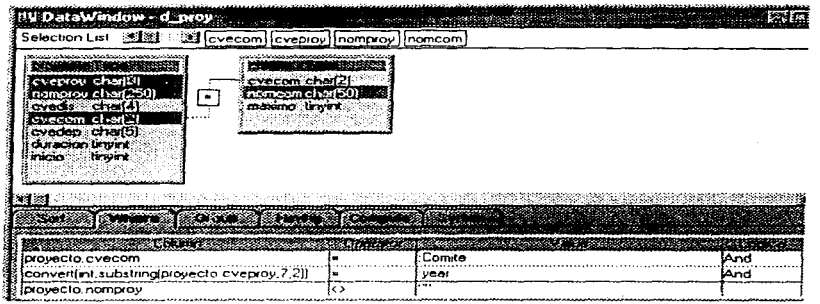
Paso 6

Creamos un ***datawindow con 2 argumentos*** **SQL Select** tipo **Tabular** con los siguientes campos: la clave del comité (*cvecom*), la clave del proyecto (*cveproy*) y el nombre del proyecto (*nomproy*) de la **tabla proyecto**, también el nombre del comité (*nomcom*) de la **tabla comite**.

Los argumentos serían los siguientes:



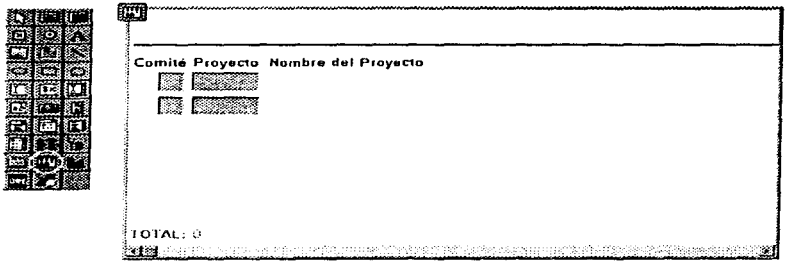
Y se usarían así en el datawindow:



Guardamos el datawindow con el nombre de **d_proy**.

Paso 7

Crear una una lista de los proyectos existentes por comité y año en una ventana. La ventana debe ser de tipo "Popup", tener asociado el menú **m_export**, con "Toolbar" Float (estas opciones se seleccionan desde las propiedades de la ventana) y con la siguiente apariencia :



El datawindow pegado es el realizado anteriormente **d_proy**.

El siguiente script para el *evento open* de la ventana es el siguiente:

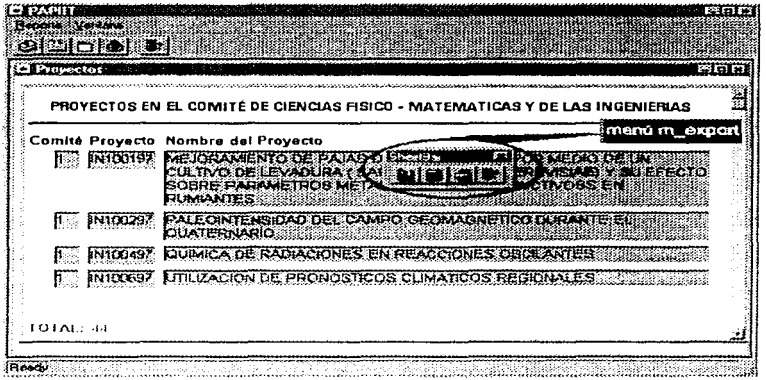
```
// NOTA: dw_1 es el nombre del objeto datawindow dentro de
// una ventana (este nombre puede ser cambiado, por ejemplo a dw_proyecto)
// Establece una conexión con el servidor
connect;
// Define el objeto de transacción a usar
dw_1.settransobject(SQLCA)
// Recupera la información
dw_1.retrieve()
// Cierra la conexión
   disconnect;
```

Guardamos la ventana con el nombre de w_proy y colocamos el siguiente script en la opción de "Información -> Proyectos" del menú principal (m_principal):

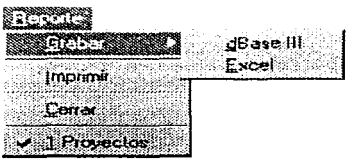
```
// Activa el cursor de reloj (espera)
setpointer(hourglass!)
// Abrimos la ventana de proyectos como una ventana hija de w_principal
// y ajustada a su tamaño (layered!)
// w_proy tendrá su propio menú (m_export)
opensheet(w_proy,w_principal,1,layered!)
```

 Prueba de la aplicación

Al oprimir el painter Run (), elegimos la opción "Información -> Proyectos" del menú principal, y veríamos algo así:



El menú m_export:

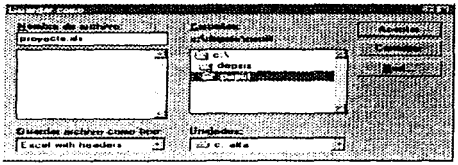


tiene las siguientes opciones: grabar la información a un archivo de dBase o Excel, o bien, imprimir los datos. Estas opciones pueden ser ejecutadas desde el menú de "Reporte" o bien desde la barra de herramientas que se muestra en "flotando" sobre la ventana.

El script que debe de ir en la opción grabar la información en Excel es el siguiente:

```
// Exportando datos a Excel
w_proy.dw_1.SaveAs("",Excel!,TRUE)
```

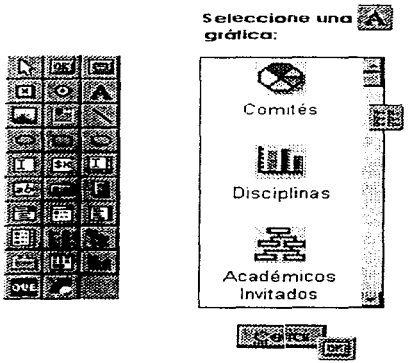
Y el resultado sería este:



La ventaja de grabar los datos en distintos formatos le da la facilidad al usuario de manipular la información con diversas herramientas para existentes para el procesamiento de datos.

Paso 8

Creamos una ventana tipo "Main" con la siguiente apariencia:



El script que debe ir en el *evento clicked* de la "lista listview" ( ) es el siguiente:

```
integer opc
// Ver que opción esta seleccionada
opc = lv_1.SelectedIndex( )
// Manda llamar una gráfica hecha en Crystal Reports
// según la opción seleccionada
If opc = 1 then run("c:\siset\comgui1.exe")
If opc = 2 then run("c:\siset\comgui2.exe")
If opc = 3 then run("c:\siset\comgui3.exe")
```

El script del *evento clicked* del botón "Cerrar" es el siguiente:

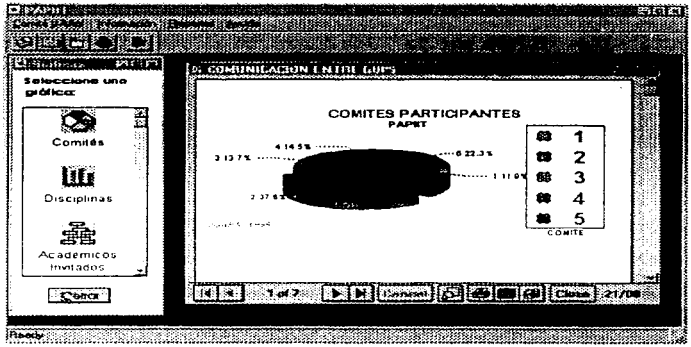
```
// Cierra la ventana
close(parent)
```

Guardamos la ventana con el nombre de **w_grafica** y colocamos el siguiente script en la opción de "Reportes -> Gráficas" del menú principal (m_principal):

```
// Activa el cursor de reloj (espera)
setpointer(hourglass!)
// Abrimos la ventana de gráficas
open(w_grafica)
```

Prueba de la aplicación

Al oprimir el painter Run (), elegimos la gráfica de Comités de la opción "Reportes -> Gráficas" del menú principal, y veríamos algo así:



Paso 9

Crear una una lista ventana tipo "Response" con la apariencia siguiente:

El script del *evento clicked* del botón Aceptar es el siguiente:

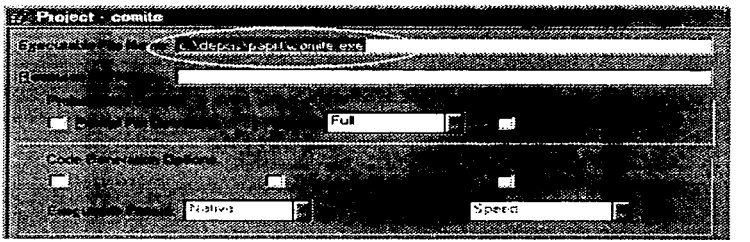
```
// Cerrar la ventana
close(parent)
```

Guardamos la ventana con el nombre de **w_about** y colocamos el siguiente script en la opción de "Ayuda -> Acerca del programa..." del menú principal (m_principal):

```
// Activa el cursor de reloj (espera)
setpointer(hourglass!)
// Abrimos la ventana créditos del programa
open(w_about)
```

Creación de un programa ejecutable

Para concluir la aplicación, crearemos un programa ejecutable de la aplicación usando el painter Project ().



Especificamos la ubicación y nombre del programa ejecutable y a continuación la opción "Build Project" del menú Design, o bien, oprimimos el painter Build () para crear el programa.

Una vez creado el programa ejecutable se puede correr independientemente de PowerBuilder (no se requiere tener PowerBuilder).

Conclusiones

Hoy en día estamos viviendo en un mundo en el que la comunicación juega un papel muy importante y es donde el avance de la tecnología ha aportado diversos medios de comunicación y difusión, Internet es un ejemplo. Y con Internet el World Wide Web o simplemente Web.

El Web durante muchos años ha sido un medio donde se difunde información, versiones de evaluación de programas de cómputo o solamente publicidad; sin embargo, el potencial más grande del Web, aún no explotado, en su totalidad, esta en los sistemas de información, los cuales permiten transmitir información de un lugar del mundo a otro en fracciones de segundo.

Con la realización de este proyecto, hemos aprendido ha analizar, planear y dirigir sistemas de información usando diferentes técnicas de programación, utilizando el Web y herramientas GUIs; así mismo, hemos adquirido los conocimientos técnicos necesarios para la construcción e implementación de dichos sistemas sobre una arquitectura Cliente/Servidor.

El sistema de información puesto en marcha en el Web, permite a los investigadores de la Universidad Nacional Autónoma de México la captura de la solicitud de ingreso al PAPIIT (Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica), con lo cual cumplimos con una de las expectativas planteadas al inicio de este trabajo.

Con el Web surgieron nuevas herramientas de gráficas de desarrollo o herramientas GUIs. Las GUIs han tenido en pocos años una gran aceptación, gracias a las ventajas que ofrecen, entre ellas, la variedad de aplicaciones que se pueden desarrollar en poco tiempo, el poder de comunicación entre ellas y la flexibilidad de su uso.

El sistema de información puesto en marcha internamente en la DGAPA desarrollado con una herramienta GUI, le permite a las personas encargadas de la administración del PAPIIT consultar a los proyectos aceptados en el programa, con lo que se cubre una de las necesidades de la institución.

Finalmente, hemos visto que en cuestión de cómputo, muchas técnicas de programación surgen día con día, y en este sentido, está en nuestras manos el estudiar y sacar provecho de éstas técnicas, con la finalidad tener una mejor visión de la evolución de sistemas de cómputo y con ello poder tomar la mejor decisión en cuanto a la creación de sistemas de información.

Hace sólo un año cuando personas "apostaron" que Internet iba a desaparecer, sin embargo, los hechos demuestran lo contrario.
Internet la red de computadoras más grande del mundo incrementa cada vez más su número de usuarios, prueba de ello son las decenas de compañías comercializadoras de Internet que incorporan a miles de usuarios a Internet día a día.

Lo anterior nos lleva al siguiente cuestionamiento: si cada día aumenta el número de usuarios en Internet, entonces va ha llegar el momento en que se sature y de lugar al surguimiento de otras alternativas que la reemplacen, o no?.

Efectivamente, es cierto que la Internet se ha extendido rápidamente y que quedan pocas direcciones IP (una dirección IP es como un número telefónico, existe uno por cada teléfono y no se puede repetir) por asignar, sin embargo, ya existe una solución, conocida como "Nueva generación de IP" o IPng. IPng va ha permitir incrementar el número de direcciones IP sin modificar las ya existentes, va ha ser escalable para redes de muchos tamaños y capacidades de transmisión:

En este sentido Internet tiene mucho futuro, y con Internet el Web, el cual es actualmente uno de los servicios de Internet que ha causado un gran impacto y día con día es mejorado con nuevas herramientas que permiten una mayor flexibilidad para el desarrollo de sistemas bajo esta tecnología por ejemplo mencionemos a Javascript y más recientemente a Java. Por lo

anterior los sistemas de información tienen un gran futuro y son susceptibles de ser mejorados y ofrecer un mejor servicio.

A corto plazo pensamos que podemos poner varios programas administrados por la DGAPA bajo el concepto WWW e igualmente podemos pensar en un sistema de consulta interno para la manipulación recibida vía Internet incorporando las nuevas tecnologías y conceptos de programación.

A mediano plazo podemos explotar todo el potencial de Internet dentro de la misma dependencia, creando una Intranet. Con la Intranet podemos hacer uso de todos los servicios de Internet aislando a esta del resto del mundo. Los servicios que podemos usar son el correo electrónico (y no el teléfono) como medio de comunicación, los grupos de discusión (newsgroups) como una forma para resolver problemas y Netscape como una forma de difusión de información interna.

A largo plazo podemos pensar en un sistema de información bajo el concepto WWW que funciona en toda la UNAM, que permita concentrar información actualizada de una manera sencilla y eficaz.

Ciclo de vida
Pressman, S. Roger
Ingeniería del software. Un enfoque práctico
McGraw-Hill
3a. Edición
Madrid, 1993

Senn, James A.
Análisis y diseño de información
McGraw-Hill
2a. Edición
México, 1994

Ciclo de vida y Visualizadores
Krol, Ed.
Conéctate al mundo de Internet. Guía y Catálogo
2a. Edición
McGraw-Hill
México, 1995

Redes de computadoras
"TCP/IP versión 6"
http://www.cis.ohio-state.edu/htbin/rfc/rfc1752.html

Tanenbaum, Andrew S.
Computers Networks
2a. Edición
Prentice Hall

Santifaller, Michael
TCP/IP NFS Interworking in a Unix Environment
Addison-Wesley

Hurriaga Velázquez Claudia y M. Germán Daniel
El conjunto de protocolos TCP/IP
Soluciones Avanzadas Año 4 No. 33
Mayo 1996

http://www/lag.itsm.mx/276146/inter9.html
http://ei.udg.es/~ramon/ib/t3/tema3.htm

Comunicaciones y Redes de Procesamiento de Datos.
González Sainz Nestor
McGraw-Hill

Bases de datos
Oracle RDBMS
Database Administrator's Guide Vol. 1 versión 7.0
pp. 1-1 a 1-7

McGoberan, D.
Sybase and SQL Server
Addison-Wesley
June 1993

Date, C. J.
Introducción a los sistemas de bases de datos
Addison-Wesley

Introducción a Sybase Transact-SQL
Guía de principiante

Mannila, Heikki & Räijär, Kari-Jouko
The Desaign of Relational Database
Addison-Wesley
Gran Bretaña, 1994

Korth F., Henrey & Sileberschatz, Abraham
Fundamentos de Bases de Datos
McGraw-Hill
México, 1990

Peñaloza B., Marcela
Selección de Sybase entre otros RDBMSs y características
de sus productos
UNAM - DGSCA - DCAA
México D.F 1994

Interfaces Gráficas de Usuario
PowerBuilder 5.0 Getting Started
Sybase, Inc.
May 1996

PowerBuilder 5.0 Project Primer
Sybase, Inc.
May 1996

Horey Jeremy y Olsen Mike
Windows 3.0
Personal Computing No. 33
1991

Visualizadores
Matuk, Javier
Como navegar por las telarañas de Internet o
el mundo a través del WWW
PC Computing Año 7 No. 81
Febrero 1995

"Tags for creating forms in HTML"
http://snowwhite.it.brighton.ac.uk/~mas/mas/courses/html/html2.html

http://smu2.cps.unizar.es/pag/5mu2_0331.html

http://www.inf.cr.uclm.../revista4/hipertac.htm

http://www.ee2.csic.es.es/manual/gopher.html

http://www.ts.es/doc/other/www_5.htm

http://www.intercom.es/intervista/tec603-1.htm

http://sun2.hubcentrlh...unith/netres4.1.1.html

http://www.licese.mx/terra/html/htmldef.html/sgml.html

http://193.145.249.23/servInf/AyudaInf/cursoHTML/curso01.htm

http://204.227.75.44.../winhtml.htm1#browsers.htm

Servidores HTTP
M. Germán, Daniel y López-Ortiz Alejandro
Instalando un servidor de WWW
Soluciones Avanzadas Año 4 No. 33
Mayo 1996

Martin Zamboni, Diego
Seguridad en los servicios de Internet
Soluciones Avanzadas Año 4 No. 33
Mayo 1996

Programación de CGI's
Open Client DB-Library / C
Reference Manual
Open Client release 10.0
August 1994

Common Gateway Interface
http://hoohoo.ncsa.uiuc.edu/cgi/intro.html

CGI overview
http://www.intergalact.com/hp/part3/overview.html

CGI scripts
http://snowwhite.it.brighton.ac.uk/~mas/mas/courses/html/html3.html

Footnotes for "CGI Made Really Easy"
http://www.jmarshall.com/easy/cgi/cgi_footnotes.html#samples

Contreras Alcalá, Felipe
CGI: La programación para HTML
Soluciones Avanzadas Año 5 No. 37
Septiembre 1996

Bibliografía Adicional
Diccionario de Computación
Freedman, Alan
Quinta Edición
McGraw-Hill
España 1993

Apéndice A

access.conf es el archivo de configuración de acceso al servidor HTTP.

```
# Inicio del archivo access.conf: Global access configuration
# Online docs at http://hoohoo.ncsa.uiuc.edu/
# I suggest you consult them; this is important and confusing stuff.

# /usr/local/etc/httpd/ should be changed to whatever you set ServerRoot to.
<Directory /usr/local/etc/httpd/cgi-bin>
Options Indexes FollowSymLinks
</Directory>

# This should be changed to whatever you set DocumentRoot to.

<Directory /usr/local/etc/httpd/htdocs>

# This may also be "None", "All", or any combination of "Indexes",
# "Includes", or "FollowSymLinks"

Options Indexes FollowSymLinks

# This controls which options the .htaccess files in directories can
# override. Can also be "None", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"

AllowOverride All

# Controls who can get stuff from this server.

<Limit GET>
order allow,deny
allow from all
</Limit>

</Directory>

# You may place any other directories you wish to have access
# information for after this one.

<Directory /home/http/httpd_1.4.2/cgi-bin/infofin>
<Limit GET>
order deny,allow
```

```
deny from all
allow from 132.248.37.
allow from 132.248.11.22

</Limit>
</Directory>

<Directory /home/http/httpd_1.4.2/cgi-bin/renovacion>
<Limit GET>
order deny,allow
deny from all
allow from 132.248.37.

</Limit>
</Directory>

#Fin del archivo access.conf
```

Apéndice B

post-query.c es el programa que decodifica las formas www y muestra en pantalla los valores en parejas (name = value) de los datos capturados.

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_ENTRIES 10000

typedef struct {
   char *name;
   char *val;
} entry;

char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);


main(int argc, char *argv[]) {
   entry entries[MAX_ENTRIES];
   register int x,m=0;
   int cl;

   printf("Content-type: text/html%c%c",10,10);

   if(strcmp(getenv("REQUEST_METHOD"),"POST")) {
      printf("This script should be referenced with a METHOD of POST.\n");
      printf("If you don't understand this, see this ");
      printf("<A
HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
      exit(1);
   }
   if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")) {
      printf("This script can only be used to decode form results. \n");
      exit(1);
   }
   cl = atoi(getenv("CONTENT_LENGTH"));
```

```
    for(x=0;cl && (!feof(stdin));x++) {
        m=x;
        entries[x].val = fmakeword(stdin,'&',&cl);
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        entries[x].name = makeword(entries[x].val,'=');
    }

    printf("<H1>Query Results</H1>");
    printf("You submitted the following name/value pairs:<p>%c",10);
    printf("<ul>%c",10);

    for(x=0; x <= m; x++)
        printf("<li> <code>%s = %s</code>%c",entries[x].name,
            entries[x].val,10);
    printf("</ul>%c",10);
}
```

Apéndice C

util.c contiene las funciones que utiliza un programa CGI para decodificar los datos capturados en una forma www. El programa CGI solo usa el código objeto de este archivo (util.o), para crearlo se debe seguir la siguiente instrucción: $gcc -o util.c

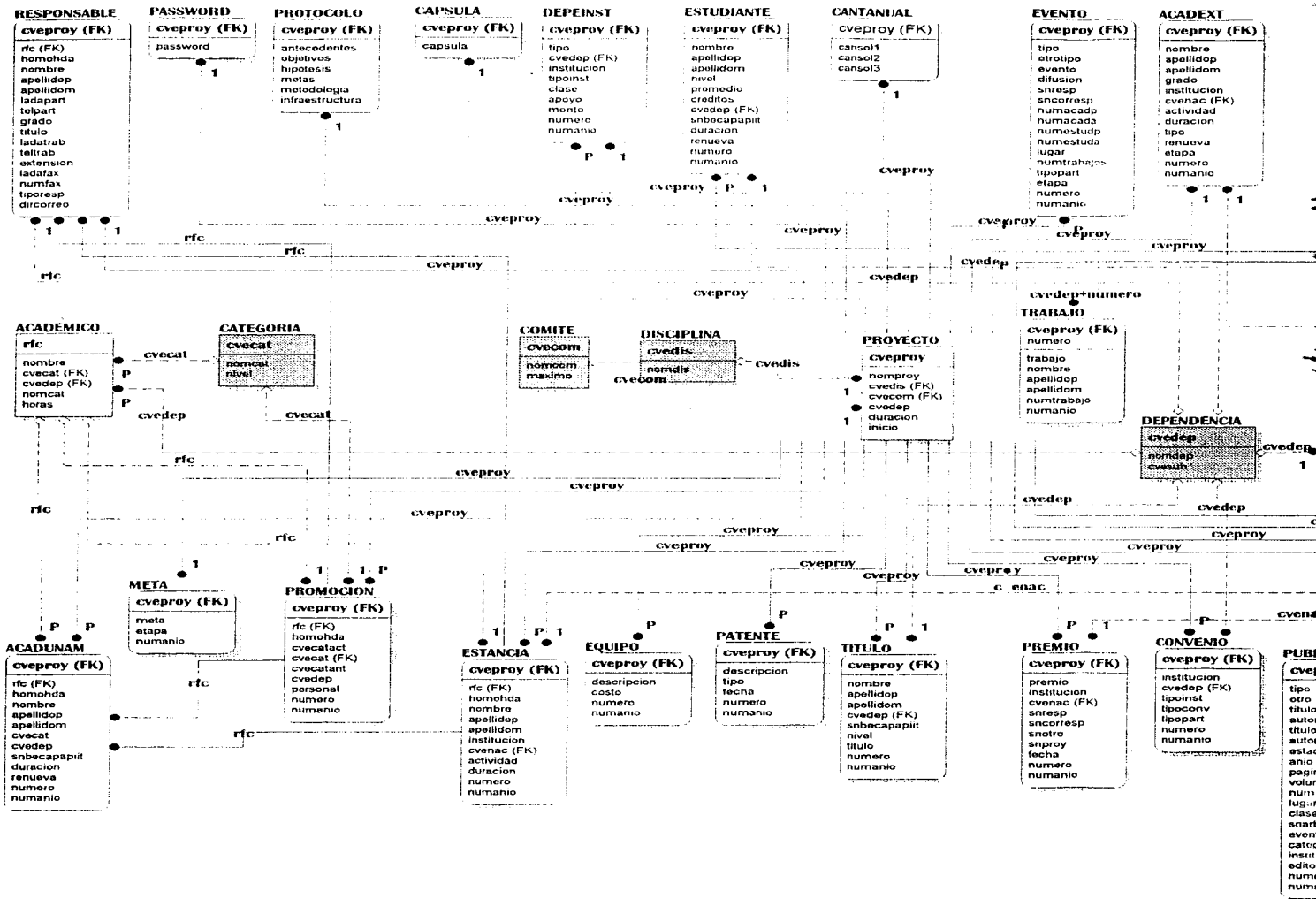
```
#include <stdio.h>

#define LF 10
#define CR 13

void getword(char *word, char *line, char stop) {
   int x = 0,y;
   for(x=0;((line[x]) && (line[x] != stop));x++)
      word[x] = line[x];
   word[x] = '\0';
   if(line[x]) ++x;
   y=0;
   while(line[y++] = line[x++]);
}

char *makeword(char *line, char stop) {
   int x = 0,y;
   char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
   for(x=0;((line[x]) && (line[x] != stop));x++)
      word[x] = line[x];
   word[x] = '\0';
   if(line[x]) ++x;
   y=0;
   while(line[y++] = line[x++]);
   return word;
}

char *fmakeword(FILE *f, char stop, int *cl) {
   int wsize;
   char *word;
   int ll;
   wsize = 102400;
   ll=0;
   word = (char *) malloc(sizeof(char) * (wsize + 1));
   while(1) {
      word[ll] = (char)fgetc(f);
      if(ll==wsize) {
         word[ll+1] = '\0';
```

```
            wsize += 102400;
            word = (char *)realloc(word,sizeof(char)*(wsize + 1));
        }
        --(*cl);
        if((word[ll] == stop) || (feof(f)) || (!(*cl))) {
            if(word[ll] != stop) ll++;
            word[ll] = '\0';
              word = (char *) realloc(word, ll + 1);
            return word;
        }
        ++ll;
    }
}

char x2c(char *what) {
    register char digit;
    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A') + 10 : (what[0] - '0'));
    digit *= 16;
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A') + 10 : (what[1] - '0'));
    return(digit);
}

void unescape_url(char *url) {
    register int x,y;
    for(x=0,y=0;url[y];++x,++y) {
        if((url[x] = url[y]) == '%') {
            url[x] = x2c(&url[y+1]);
            y+=2;
        }
    }
    url[x] = '\0';
}

void plustospace(char *str) {
    register int x;
    for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
}

int rind(char *s, char c) {
    register int x;
    for(x=strlen(s) - 1;x != -1; x--)
        if(s[x] == c) return x;
    return -1;
}

int getline(char *s, int n, FILE *f) {
    register int i=0;
    while(1) {
        s[i] = (char)fgetc(f);
```

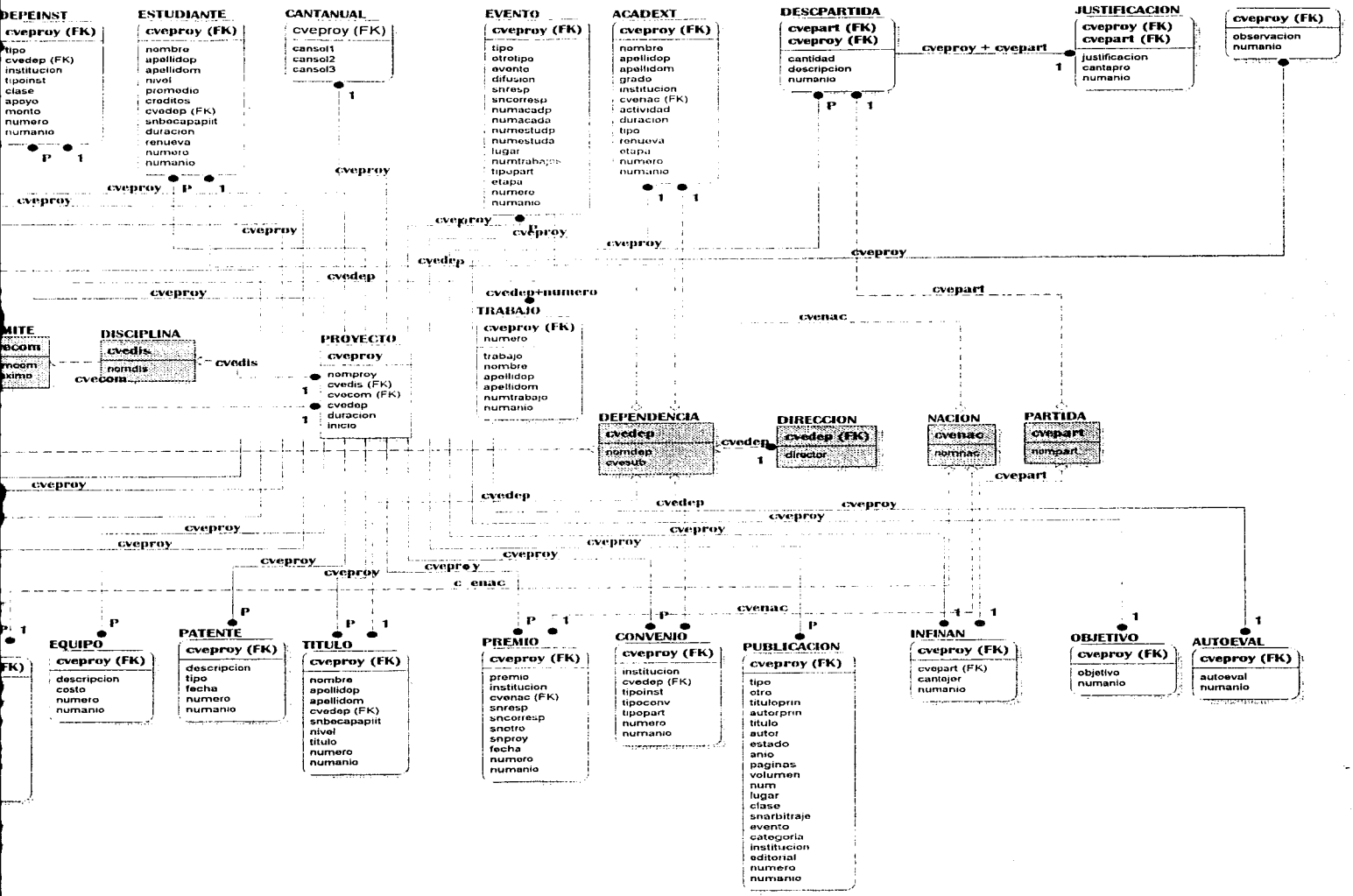
```
        if(s[i] == CR)
            s[i] = fgetc(f);
        if((s[i] == 0x4) || (s[i] == LF) || (i == (n-1))) {
            s[i] = '\0';
            return (feof(f) ? 1 : 0);
        }
        ++i;
    }
}

void send_fd(FILE *f, FILE *fd)
{
    int num_chars = 0;
    char c;
    while (1) {
        c = fgetc(f);
        if(feof(f))
            return;
        fputc(c,fd);
    }
}

int ind(char *s, char c) {
    register int x;
    for(x = 0;s[x];x++)
        if(s[x] == c) return x;
    return -1;
}

void escape_shell_cmd(char *cmd) {
    register int x,y,l;
    l = strlen(cmd);
    for(x = 0;cmd[x];x++) {
        if(ind("&;`'\"|*?~<>^()[]{}$\\",cmd[x]) != -1){
            for(y = l+1;y > x;y--)
                cmd[y] = cmd[y-1];
            l++; /* length has been increased */
            cmd[x] = '\\';
            x++; /* skip the character */
        }
    }
}
```

Apéndice D

El diagrama entidad/relación (diagrama E/R) resultante del análisis y diseño de la base de datos a utilizar en el programa PAPIIT se muestra en las siguientes páginas.



BASE DE DATOS DEL SISTEMA PARA PAPIIT



MA PARA PAPIIT

Fecha de actualización: 4 de junio de 1996